A Hoare Logic for Linear Systems

Rob Arthan, Ursula Martin and Paulo Oliva

School of Electronic Engineering and Computer Science
Queen Mary, University of London
London E1 4NS, UK

Abstract. We consider reasoning about linear systems expressed as block diagrams that give a graphical representation of a system of differential equations or recurrence equations. We use the notion of additive relation borrowed from homological algebra to give a convenient framework in which all diagrams have a semantic value. We give a sound system of Hoare-style rules for the block diagram constructors that singles out a tractable subset of the block diagram language in which all diagrams represent total functions. We show these rules in action on some simple examples from a variety of applications domains.

Keywords: Hoare logic; formal verification; linear systems; control systems

1. Introduction

Linear systems, e.g., systems of linear differential or difference equations, are much used in designing control systems in avionics and in many other areas of engineering and science. In the design process, a top level continuous model is often refined into a hybrid model involving continuous and discrete elements, the discrete aspects of the model then providing the requirements for a formal software development process.

Widely used tools such as Simulink allow systems to be expressed as a signal flow graph formed by wiring together primitive components to form subsystems in a hierarchic fashion. Such tools support validation via simulation and numerical calculation, but not via formal reasoning and mechanised proof. As indicated by the full recognition for formal methods in the forthcoming update to DO-178, the standard used by certification authorities such as FAA, EASA and Transport Canada to approve all commercial software-based aerospace systems, there is a growing demand in safety-critical applications for verification via formal, machine-checked reasoning at all levels of the design process. This demand is not met by existing technology for continuous and hybrid systems.

This paper proposes an approach that addresses some aspects of this shortfall, while remaining close to engineering practice, by annotating block diagrams with assertions, and using rules of inference to propagate reasoning about assertions through the diagram. This is in the spirit of Hoare Logic [Hoa69], which is the basis of many successful approaches to software verification [Jon03]. The approach should scale well: we can reason about both larger components and primitive blocks in the same framework. As we shall see, the

Correspondence and offprint requests to: Rob Arthan, School of Electronic Engineering and Computer Science, Queen Mary, University of London, London E1 4NS, UK. e-mail: rda@eecs.qmul.ac.uk

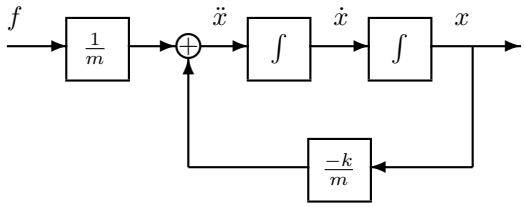


Fig. 1. Block Diagram

approach also has good potential for automation, while still giving the user control over the structure of specifications and proofs.

As an example, Figure 1 represents a mechanical system in which a force f acts on a cart of mass m attached to a wall by a spring with spring constant, k. It is a graphical representation of the following differential equation:

$$m\ddot{x}(t) + kx(t) = f(t). \tag{1}$$

Thinking of the force function $f(t)$ as the input to the system and the position function $x(t)$ as the output leads to the intensional view of the system depicted in the figure. As we will see this lets us reason about the system in terms of pre- and post-conditions describing the behaviour of the system and its subsystems.

The programme has three elements:

- We need a **semantics** for block diagrams and assertions about them. Block diagrams define what are called additive relations between inputs and outputs. Our assertions define classes of states at various points in the diagram.
- A Hoare style **logic** for this semantics, to enable the propagation of assertions through the diagram. The **axioms** consist of decidable primitives, and we give **proof rules** for the various ways of constructing new diagrams from old, allowing one to prove assertions at chosen points in a complex diagram.
- The usual **tools** for supporting linear systems and block diagrams are numeric, implemented in systems such as Simulink. For this work we need to add a computational logic engine to enable development of Hoare-style proofs and to support formal machine-checked proof of verification conditions derived during the proof process.

In [AMMO07] we gave a form of Hoare logic for a block diagram language in which the semantic domains comprise vector spaces and linear transformations. That logic was unusual in having non-trivial semantic side-conditions, which, however, in some applications areas could be expressed in a decidable theory. In the sequel, our goal is to reconcile the work of [AMMO07] with the methodology for defining Hoare logics given in [AMMO09]. We do this by assuming additional mathematical structure: we now work with Banach spaces; we also work with an assertion language that uses the Banach space norm to express pre- and post-conditions with a specified error tolerance.

The work of [AMMO07] and the present paper were inspired by earlier work [BHM03], where explicit inference rules expressed in terms of trigonometric functions were given for a restricted class of linear functions arising in control engineering. That in turn was motivated by work of the first author and others on ClawZ [ACOS00], which supports reasoning in ProofPower about block diagrams via a translation into the Z specification language. ClawZ is being used with good results by QinetiQ in avionics and other safety-critical applications.

The structure of the sequel is as follows:

Section 2 recalls the main mathematical notions that inform our approach; we review additive relations, a generalisation of the notion of a linear transformation between vector spaces, and show how these provide a convenient semantic domain for systems expressed using feedback loops that denote linear equational constraints. We show the correspondence between the loop construct of [AMMO07] and the trace operator of the traced monoidal categories used in [AMMO09]. We recall the concept of a Banach space and of a bounded operator and give a sufficient condition for the trace of a bounded operator to be itself a bounded operator.

Section 3 defines the notion of a signal flow graph and introduces the block diagram language that is the

"programming language" of our Hoare logic. The language is parametrised by a structure providing a system of named Banach spaces and bounded operators that can be tailored to the needs of a particular application. We then observe that the language can be viewed as a derived syntax for a language based on the traced monoidal category constructors as used in [AMMO09].

Section 4 introduces the Hoare logic for our block diagram language and gives the proof that it is sound. While the general rule for the trace construct still involves a semantic side-condition, we show that a less general rule that is very useful in practice allows the side-condition to be expressed in our assertion language.

Section 5 considers examples from several applications domains: the spring-and-cart system sketched above, linear filters and a subsystem of the well-known steam-boiler problem.

Section 6 discusses related work and future directions.

2. Mathematical Preliminaries

2.1. The category of vector spaces

We are interested in continuous systems whose inputs, outputs and internal states are physical quantities such as positions in space, voltages or rates of flow. These quantities will be represented by elements of vector spaces. For example, in systems like that of Figure 1, the positions, velocities, accelerations and forces might be real-valued functions of time, i.e., members of the set $V = \mathbb{T} \to \mathbb{R}$, where $\mathbb{T}$ is some set representing times; we obtain valuable algebraic structure by regarding V as a real vector space, scaling and adding functions pointwise: $(af)(\tau) = a(f(\tau))$, $(f + g)(\tau) = f(\tau) + g(\tau)$ for $a \in \mathbb{R}$, $\tau \in \mathbb{T}$.

While much of what follows holds for vector spaces over other fields of scalars, we will restrict our attention in the sequel to the important case when the field of scalars is the field of real numbers. The category $\mathcal{V}$ of real vector spaces has the class of real vector spaces for its class of objects. Given vector spaces V and W, the morphisms between V and W in $\mathcal{V}$ are the *linear transformations*, i.e., the functions $f : V \to W$ that commute with the vector space operations: $f(a\mathbf{v} + b\mathbf{w}) = af(\mathbf{v}) + bf(\mathbf{w})$ for $\mathbf{v}, \mathbf{w} \in V$, $a, b \in \mathbb{R}$. (We adopt the convention of using bold font for vectors in a vector space unless custom dictates otherwise, e.g., for spaces of real-valued functions, as in the previous paragraph).

$\mathcal{V}$ has an initial object given by the vector space $0 = \{\mathbf{0}\}$ and products given by equipping the set-theoretic product with vector space operations defined component-wise. We write $\pi_i : X_1 \times \ldots \times X_n \to X_i$, $i = 1, \ldots, n$, for the projection $(\mathbf{x}_1, \ldots, \mathbf{x}_n) \mapsto \mathbf{x}_i$. If $f, g : V \to W$ are linear transformations, then so is their sum, $f + g$, defined point-wise: $(f + g)(\mathbf{v}) = f(\mathbf{v}) + g(\mathbf{v})$. This makes $V \to W$ into an abelian group, the zero element being the linear transformation $0_{VW} = \mathbf{v} \mapsto \mathbf{0}$. This addition satisfies the naturality conditions required to make $\mathcal{V}$ into an additive category. In particular, this implies that finite products in $\mathcal{V}$ are also finite sums and that a linear transformation $f = V_1 \times \ldots \times V_m \to W_1 \times \ldots \times W_n$ has a block matrix representation (f_{ij}) where $f_{ij} = (\iota_i; f; \pi_j)$, $\iota_i : V_i \to V_1 \times \ldots \times V_m$ being the injection of the summand into the sum, $\mathbf{x}_i \mapsto (\mathbf{0}, \ldots, \mathbf{0}, \mathbf{x}_i, \mathbf{0}, \ldots, \mathbf{0})$.

So that we can write n-tuples as row vectors in the usual way, we will write matrices on the right of the vectors on which they act. Because of this, and because it reads well in connection with the left-to-right composition operator ';', we will often write functions, and particularly linear transformations on the right of their argument. So, for example, we would write the calculations to verify the naturality conditions for the addition on the hom-sets of the additive category $\mathcal{V}$ as follows:

$$\mathbf{u}(p; (f + g)) = \mathbf{u}p(f + g) = \mathbf{u}pf + \mathbf{u}pg = \mathbf{u}(p; f) + \mathbf{u}(p; g) = \mathbf{u}((p; f) + (p; g))$$

$$\mathbf{v}((f + g); q) = (\mathbf{v}(f + g))q = (\mathbf{v}f + \mathbf{v}g)q = \mathbf{v}fq + \mathbf{v}gq = \mathbf{v}(f; q) + \mathbf{v}(g; q) = \mathbf{v}((f; q) + (g; q))$$

where $\mathbf{v} \in V$, $\mathbf{u} \in U$, $p : U \to V$, $f, g : V \to W$ and $q : W \to X$.

2.2. Additive relations

The notion of additive relation was designed for reasoning about the "diagram-chasing" methods of homological algebra [Mac75]. Additive relations provide a pleasant framework for the semantics of the diagrammatic notations of interest here.

We work with systems modelled as (binary) relations between an input space and an output space. The input-output relation will be the solution to some system of equations represented in a diagrammatic notation. The input-output relation may be partial (i.e., there may be no solutions for some inputs) and may be non-deterministic (i.e., there may be inputs for which there is more than one solution). The solutions will always be *additive relations*, as defined below. In practice, one is mainly concerned with systems that are total and functional, but it is convenient to work in a setting where all systems of equations have a solution.

Before defining additive relations, let us agree on some notation. We write $X \leftrightarrow Y$ for the set $\mathbb{P}(X \times Y)$ of all relations between the set X and the set Y. We use underlining to highlight infix use of relations, i.e., when $r : X \leftrightarrow Y$, $x \mathrel{\underline{r}} y$ means $(x, y) \in r$ If r and s are relations, $r; s$ denotes the relational composition, r followed by s, so that $x \mathrel{\underline{(r;s)}} y$ iff. there is a z with $x \mathrel{\underline{r}} z$ and $z \mathrel{\underline{s}} y$. If $r : X \leftrightarrow Y$, $r^{-1} : Y \leftrightarrow X$ is the relational inverse of r, defined by taking $x \mathrel{\underline{r^{-1}}} y$ iff. $y \mathrel{\underline{r}} x$. 1_X is the identity function on the set X. We write Ar for the image of a set A under the relation r when confusion with other notations is unlikely. So if $r : X \leftrightarrow Y$, then $\mathsf{dom}(r) = Yr^{-1}$ is the domain of r and $\mathsf{ran}(r) = Xr$ is its range.

Definition 2.1. Let V and W be vector spaces. An **additive relation** between V and W is any subspace of the cartesian product $V \times W$. Equivalently, an additive relation is any non-empty relation, $r : V \leftrightarrow W$, such that if $\mathbf{v}_1, \mathbf{v}_2 \in V$, $\mathbf{w}_1, \mathbf{w}_2 \in W$, and $\mathbf{v}_1 \mathrel{\underline{r}} \mathbf{w}_1$ and $\mathbf{v}_2 \mathrel{\underline{r}} \mathbf{w}_2$, then also $a\mathbf{v}_1 + b\mathbf{v}_2 \mathrel{\underline{r}} a\mathbf{w}_1 + b\mathbf{w}_2$ for any $a, b \in \mathbb{R}$.

For example, (the graph of) any linear transformation $f : V \to W$ is an additive relation between V and W. The kernel of a linear transformation can be thought of as giving a uniform measure of the information it loses. The same idea applies to additive relations, for which we also have a dual notion: the indeterminacy as defined below gives a uniform measure of the non-determinism of the relation.

Definition 2.2. If $r : V \leftrightarrow W$ is an additive relation, the **kernel** and **indeterminacy** of r are defined by $\mathsf{ker}(r) = \{\mathbf{v} : V \mid \mathbf{v} \mathrel{\underline{r}} 0\}$ and $\mathsf{ind}(r) = \{\mathbf{w} : W \mid 0 \mathrel{\underline{r}} \mathbf{w}\}$, respectively.

Lemma 2.3. If $r : V \leftrightarrow W$ is an additive relation, then $\mathsf{dom}(r)$ and $\mathsf{ker}(r)$ are subspaces of V and $\mathsf{ran}(r)$ and $\mathsf{ind}(r)$ are subspaces of W. Moreover, the inverse relation, r^{-1} is also an additive relation and one has $\mathsf{dom}(r^{-1}) = \mathsf{ran}(r)$, $\mathsf{ran}(r^{-1}) = \mathsf{dom}(r)$, $\mathsf{ker}(r^{-1}) = \mathsf{ind}(r)$ and $\mathsf{ind}(r^{-1}) = \mathsf{ker}(r)$.

Proof. Routine. □

If X and Y are subsets of a vector space V, we write $X + Y$ for the set $\{\mathbf{z} \mid \exists \mathbf{x} : X, \mathbf{y} : Y \bullet \mathbf{z} = \mathbf{x} + \mathbf{y}\}$. If $\mathbf{x} \in V$, we write $\mathbf{x} + X$ for $\{\mathbf{x}\} + X$. Recall that if X happens to be a subspace of V, then sets of the form $\mathbf{x} + X$ are referred to as **cosets** of the subspace X and the collection of all cosets forms the quotient vector space V/X. The following lemma shows that an additive relation relates cosets of its kernel to cosets of its indeterminacy.

Lemma 2.4. Let $r : V \leftrightarrow W$ be an additive relation and assume $\mathbf{v} \mathrel{\underline{r}} \mathbf{w}$, then

$$\begin{aligned} \{\mathbf{v}_1 : V \mid \mathbf{v}_1 \mathrel{\underline{r}} \mathbf{w}\} &= \mathbf{v} + \mathsf{ker}(r) \\ \{\mathbf{w}_1 : W \mid \mathbf{v} \mathrel{\underline{r}} \mathbf{w}_1\} &= \mathbf{w} + \mathsf{ind}(r). \end{aligned}$$

Proof. Assume $\mathbf{v}_1 \mathrel{\underline{r}} \mathbf{w}$, then by the additivity of r, $\mathbf{v}_1 - \mathbf{v} \mathrel{\underline{r}} \mathbf{w} - \mathbf{w} = 0$, so $\mathbf{v}_1 - \mathbf{v} \in \mathsf{ker}(r)$, i.e., $\mathbf{v}_1 \in \mathbf{v} + \mathsf{ker}(r)$, proving the left-to-right direction of the first equation. The rest is similar. □

Example 2.5. Assume $V_1 \subseteq V$ and $W_1 \subseteq W$ are subspaces and $f : V_1 \to U$ and $g : W_1 \to U$ are linear transformations for some vector space U, then $(f; g^{-1})$ defines an additive relation between V and W.

The following proposition shows that any additive relation arises from the construction of the above example.

Proposition 2.6. Let $r : V \leftrightarrow W$ be an additive relation, then there is a linear transformation $g : \mathsf{ran}(r) \to U$ such that $r = (f; g^{-1}) : \mathsf{dom}(r) \leftrightarrow \mathsf{ran}(r)$, where $U = \mathsf{dom}(r)/\mathsf{ker}(r)$ and $f : \mathsf{dom}(r) \to U$ is the natural projection onto the quotient space U.

Proof. Recall that the natural projection $f : \mathsf{dom}(r) \to U$ is defined by $f(\mathbf{v}) = \mathbf{v} + \mathsf{ker}(r)$. If $\mathbf{w} \in \mathsf{ran}(r)$, then, by definition, there is a $\mathbf{v}_1$ such that $\mathbf{v}_1 \mathrel{\underline{r}} \mathbf{w}$, and, by Lemma 2.4, $\{\mathbf{v} \mid \mathbf{v} \mathrel{\underline{r}} \mathbf{w}\} = \mathbf{v}_1 + \mathsf{ker}(r)$. It follows that we may define $g : \mathsf{ran}(r) \to U$ by $g(\mathbf{w}) = \{\mathbf{v} \mid \mathbf{v} \mathrel{\underline{r}} \mathbf{w}\}$. From these definitions, it is easy to check, using

Lemma 2.4, that $r = (f; g^{-1})$. □

For a more concrete interpretation of the lemma, suppose that $\mathsf{dom}(r)$ and $\mathsf{ran}(r)$ are given explicitly as the kernels of linear transformations $p : V \to C$ and $q : W \to D$ say. If one takes $f^* = f \times p \times 0 : V \times V \times V \to U \times C \times D$ and $g^* = g \times 0 \times q : W \times W \times W \to U \times C \times D$, one has for any $\mathbf{v} \in V$ and any $\mathbf{w} \in W$, $\mathbf{v}\ \underline{r}\ \mathbf{w}$ iff. $\mathbf{v}f^* = \mathbf{w}g^*$. To be even more concrete, if V and W are finite-dimensional spaces of row vectors, say $V = \mathbb{R}^m$ and $W = \mathbb{R}^n$, there are an $m \times p$ matrix, $\mathbf{F}$, and an $n \times p$ matrix, $\mathbf{G}$, for some p, such that $\mathbf{v}\ \underline{r}\ \mathbf{w}$ iff. $\mathbf{vF} = \mathbf{wG}$.

The relational composition of two additive relations is additive. Moreover the (graph of the) identity function $1_V : V \to V$ is an additive relation, so we have a category, which we call $\mathcal{V}_+$, whose objects are vector spaces and whose morphisms are additive relations. For $r : X \leftrightarrow Y$, $s : V \leftrightarrow W$, we define $r \times s : X \times V \leftrightarrow Y \times W$ by $(\mathbf{x}, \mathbf{v})\ \underline{r \times s}\ (\mathbf{y}, \mathbf{w})$ iff $\mathbf{x}\ \underline{r}\ \mathbf{y}$ and $\mathbf{v}\ \underline{s}\ Vw$. The cartesian product of vector spaces therefore serves as a product in $\mathcal{V}_+$: for $r_i : X \leftrightarrow Y_i$, $i = 1, 2$, we define the pairing $(r_1, r_2) : X \to Y_1 \times Y_2$ by $\mathbf{x}\ \underline{(r_1, r_2)}\ (\mathbf{y}_1, \mathbf{y}_2)$ iff $\mathbf{x}\ \underline{r_i}\ \mathbf{y}_i$, $i = 1, 2$. (Happily, if the r_i are linear transformations, this can equally well be viewed as a block matrix decomposition).

The following method for constructing new additive relations from old generalises the familiar pointwise sum of linear transformations. Note that this is *not* the direct sum of subspaces of $V \times W$.

Definition 2.7. Let V and W be real vector spaces.

- If $r, s : V \leftrightarrow W$ is an additive relation, then the **sum** of r and s, written $r + s$, is defined by taking $\mathbf{v}\ \underline{r + s}\ \mathbf{w}$ iff there exist $\mathbf{w}_1, \mathbf{w}_2$ such that $\mathbf{v}\ \underline{r}\ \mathbf{w}_1$, $\mathbf{v}\ \underline{s}\ \mathbf{w}_2$ and $\mathbf{w} = \mathbf{w}_1 + \mathbf{w}_2$.

We may define various other derived constructions, e.g., scalar multiplication: if $a \in \mathbb{R}$ and $r : V \leftrightarrow W$ is an additive relation, we write ar for the composite: $((\mathbf{v} : V \mapsto a\mathbf{v}); r)$. In particular, taking $a = -1$, we have negation: $\mathbf{x}\ \underline{-r}\ \mathbf{y}$ iff. $\mathbf{x}\ \underline{r}\ -\mathbf{y}$, and then, as usual we write $r - t$ for $r + -t$. Note that sum is not a group operation for additive relations in general: $r - r = V \times \mathsf{ind}(r) \neq 0_{VW}$ unless r is functional.

2.3. Traced Monoidal Categories of Vector Spaces and Banach Spaces

We now study two important methods for constructing new additive relations from old that abstract the idea of finding the solution of certain systems of equations. Both constructions will correspond to a feedback loop in a diagrammatic representation of a system.

Definition 2.8. Let V, W and Z be real vector spaces.

- If $r : V \leftrightarrow W$ and $t : W \leftrightarrow V$ are additive relations, then the **loop** of r and t, written $\mathsf{loop}\,(r, t)$, is defined by taking $\mathbf{v}\ \underline{\mathsf{loop}\,(r, t)}\ \mathbf{w}$ iff there exists $\mathbf{v}_1$ such that $\mathbf{w}\ \underline{t}\ \mathbf{v}_1$ and $(\mathbf{v} + \mathbf{v}_1)\ \underline{r}\ \mathbf{w}$.
- if $q : V \times Z \leftrightarrow W \times Z$ is an additive relation, then the **trace** of q, written $\mathsf{trace}\,(q)$, is defined by taking $\mathbf{v}\ \underline{\mathsf{trace}\,(q)}\ \mathbf{w}$ iff there exists $\mathbf{z}$ such that $(\mathbf{v}, \mathbf{z})\ \underline{q}\ (\mathbf{w}, \mathbf{z})$.

It is easy to show that the loop and trace constructions yield additive relations. The following theorem shows that the loop operation, the trace operation and the relational inverse are interdefinable.

Theorem 2.9. If $r : V \leftrightarrow W$, $t : V \leftrightarrow W$ and $u : V \times Z \leftrightarrow W \times Z$ are additive relations, then:

$$\textit{(i)}\quad \mathsf{loop}\,(r, t) = (r^{-1} - t)^{-1}$$

$$\textit{(ii)}\quad r^{-1} = \begin{pmatrix} (0_{WV}, 1_W) & : & W \to V \times W; \\ \mathsf{loop}\,(1_{V\times W}, 1_{V\times W} - (\pi_1; r; (0_{WV}, 1_W))) & : & V \times W \leftrightarrow V \times W; \\ \pi_1 & : & V \times W \to V \end{pmatrix}$$

$$\textit{(iii)}\quad \mathsf{trace}\,(u) = \begin{pmatrix} (1_V, 0_{VZ}) & : & V \to V \times Z; \\ \mathsf{loop}\,(u, (\pi_2; (0_{ZW}, 1_Z))) & : & V \times Z \leftrightarrow W \times Z; \\ \pi_1 & : & W \times Z \to W \end{pmatrix}$$

$$\textit{(iv)}\quad \mathsf{loop}\,(r, t) = \mathsf{trace} \begin{pmatrix} \pi_1 + \pi_2 & : & V \times V \to V; \\ r & : & V \leftrightarrow W; \\ (1, t) & : & W \leftrightarrow W \times V \end{pmatrix}.$$

Proof. For *(i)*, one has that $\mathbf{v}\ \underline{\mathsf{loop}\,(r,t)}\ \mathbf{w}$, iff. $\mathbf{v}+\mathbf{v}_1\ \underline{r}\ \mathbf{w}$ and $\mathbf{w}\ \underline{t}\ \mathbf{v}_1$ for some $\mathbf{v}_1$, iff., $\mathbf{w}\ \underline{r^{-1}}\ \mathbf{v}+\mathbf{v}_1$ and $\mathbf{w}\ \underline{t}\ \mathbf{v}_1$, iff. $\mathbf{w}\ \underline{r^{-1}-t}\ \mathbf{v}$, iff. $\mathbf{v}\ \underline{(r^{-1}-t)^{-1}}\ \mathbf{w}$. For *(ii)*, note that if $q : X \to X$ is an additive relation, then $1_X^{-1} - (1_X - q) = q$. So applying *(i)*, we find:

$$\begin{array}{rcl}(0_{WV}, 1_W); \mathsf{loop}\,(1_{V\times W}, 1_{V\times W} - (\pi_1; r; (0_{WV}, 1_W)))\,; \pi_1 &=& (0_{WV}, 1_W); (\pi_1; r; (0_{WV}, 1_W))^{-1}; \pi_1\\ &=& ((0_{WV}, 1_W); (0_{WV}, 1_W)^{-1}); r^{-1}; (\pi_1^{-1}; \pi_1)\\ &=& 1_W; r^{-1}; 1_V\\ &=& r^{-1}.\end{array}$$

Finally, *(iii)* and *(iv)* are easy consequences of the definitions (note $\pi_1 + \pi_2$ in *(iv)* is $(\mathbf{v}_1, \mathbf{v}_2) \mapsto \mathbf{v}_1 + \mathbf{v}_2$). □

Note that Theorem 2.9 and Proposition 2.6 imply that if a subcategory $\mathcal{C}$ of $\mathcal{V}_+$ is closed under either loop or trace and includes all linear transformations, then $\mathcal{C} = \mathcal{V}_+$.

The trace operator makes the category of additive relations into what is known as a symmetric traced monoidal category: i.e., a category equipped with a bifunctor, in this case the cartesian product $\_ \times \_$, such that with any morphism $f : X \times Z \to Y \times Z$ there is an associated morphism $\mathsf{trace}\,(f) : X \to Y$, this structure being subject to a collection of naturality conditions (see [JSV96] for full details, but a detailed knowledge of the theory of traced monoidal categories is not required to understand the Hoare logics that we will present in this paper). The loop operator arises naturally in the semantics of the block diagram notation discussed below. Since the two are interdefinable we can transfer results on traced monoidal categories to block diagrams and their semantics. In particular, as we will see, we can apply the methods of [AMMO09] to construct Hoare logics in a systematic way.

Recall that a *normed vector space* (or just *normed space*) is a vector space V equipped with a function $||\_|| : V \to \mathbb{R}_{\geq 0}$ satisfying the following axioms:

- Triangle inequality: $\forall \mathbf{v}\ \mathbf{w}\cdot\ ||\mathbf{v}+\mathbf{w}|| \leq ||\mathbf{v}|| + ||\mathbf{w}||$
- Scaling law: $\forall a\ \mathbf{v}\cdot\ ||a\mathbf{v}|| = |a| \cdot ||\mathbf{v}||$
- Positive definiteness: $\forall \mathbf{v}\cdot\ ||\mathbf{v}|| = 0 \Leftrightarrow \mathbf{v} = \mathbf{0}$.

The most familiar example is the euclidean norm on the finite-dimensional space $\mathbb{R}^n$ for $n \in \mathbb{N}$, with the norm given by $||(x_1, \ldots, x_n)|| = \sqrt{x_1^2 + \ldots + x_n^2}$. But there are many others, e.g., the maximum norm on $\mathbb{R}^n$ given by $||(x_1, \ldots, x_n)|| = \mathsf{max}\{x_1, \ldots, x_n\}$ and the Manhattan taxi-cab norm given by given by $||(x_1, \ldots, x_n)|| = |x_1| + \ldots + |x_n|$.

A Cauchy sequence in a normed space is a sequence $\mathbf{v}_n$ of vectors such that for all $\varepsilon > 0$, there is an N such that $||\mathbf{v}_n - \mathbf{v}_m|| < \varepsilon$ whenever $N < m < n$. A Banach space is a normed space which is *complete* in the sense that every Cauchy sequence converges. Some examples of Banach spaces are:

- any finite-dimensional normed space,
- the space $\mathsf{B}(X)$ of all bounded real-valued functions on a set X under the norm defined by $||f|| = \mathsf{sup}\{|f(x)| \mid x \in X\}$,
- for any $t \in \mathbb{R}$, the space $\mathcal{L}_1[0,t]$ of equivalence classes of Lebesgue-integrable functions $f : [0,t] \to \mathbb{R}$, f and g being equivalent iff $\int_{[0,t]} |f-g| = 0$, under the norm defined by $||f|| = \int_{[0,t]} |f|$. (Here we follow custom by writing f both for a function and for its equivalence class).

Recall that a linear transformation $f : V \to W$ between normed spaces is said to be *bounded* or a *bounded operator* if there is a $b \geq 0$ such that for all $\mathbf{v} \in V$, $||f(\mathbf{v})||_W \leq b||\mathbf{v}||$. If such a b exists, there is a least such b and that is called the norm, $||f||$ of the bounded operator. This makes the vector space of bounded operators from V to W into a normed space itself. It can be shown that boundedness is equivalent to continuity with respect to the norm topology.

An example of a bounded operator on the space $\mathcal{L}_1[0,t]$ is the definite integral operator $\int$ which features in the example of Figure 1. $\int : \mathcal{L}_1[0,t] \to \mathcal{L}_1[0,t]$ is defined by:

$$(\textstyle\int f)(x) \quad = \quad \int_{[0,x]} f \quad = \quad \int_{y=0}^{x} f(y)dy.$$

For this operator, we have:

$$\begin{array}{rclcrcl} ||\int f|| & = & \int_{x=0}^{t} |(\int f)(x)|dx & = & \int_{x=0}^{t} |\int_{y=0}^{x} f(y)dy|dx & \leq & \int_{x=0}^{t}\int_{y=0}^{x} |f(y)|dydx \\ & \leq & \int_{x=0}^{t}\int_{y=0}^{t} |f(y)|dydx & = & \int_{x=0}^{t} ||f||dx & = & t||f|| \end{array}$$

so $\int$ is bounded and $||\int|| \leq t$. For small positive ε, define a function g_ε on $[0,t]$ as follows:

$$g_\varepsilon(x) = \begin{cases} \frac{1}{\varepsilon} & \text{for } 0 \leq x \leq \varepsilon \\ 0 & \text{for } \varepsilon < x \leq t. \end{cases}$$

Then $||g_\varepsilon|| = 1$. Hence, for $\varepsilon \leq x \leq t$, we have $(\int g_\varepsilon)(x) = 1$. Therefore $||\int g_\varepsilon|| \geq t - \varepsilon$ implying that $||\int|| \geq t$. So we conclude that in $\mathcal{L}_1[0,t]$ we have $||\int|| = t$. One finds by a similar calculation that $||(\int;\int)|| = \frac{t}{2}$.

We write $\mathcal{B}$ for the category of Banach spaces and bounded operators and $\mathcal{B}_+$ for the category of Banach spaces and arbitrary additive relations. If V and W are Banach spaces, then $V \times W$ is again a Banach space under the norm defined by $||(\mathbf{v},\mathbf{w})|| = \sqrt{||\mathbf{v}||^2 + ||\mathbf{w}^2||}$. This gives a product functor for both $\mathcal{B}$ and $\mathcal{B}_+$. $\mathcal{B}_+$ is a traced monoidal category under the trace operation $\mathsf{trace}\,(\_)$ defined by existential quantification, but $\mathcal{B}$ is not: if μ is a bounded operator, $\mathsf{trace}\,(\mu)$ is an additive relation that is neither functional nor total in general. Since in practice, systems are often required to be deterministic and well-defined for all inputs, our goal now is to identify sufficient conditions for $\mathsf{trace}\,(\mu)$ to be a bounded operator.

In fact, there are many alternative ways of defining a norm on the product that are all equivalent in the sense that if $||\_||_1$ and $||\_||_2$ are two such norms, then there are $0 < s,t \in \mathbb{R}$ such that for any $\mathbf{x} \in V \times W$, we have $||\mathbf{v}||_1/t \leq ||\mathbf{v}||_2 \leq s||\mathbf{v}||_1$ In particular, if $0 < \varepsilon_1, \varepsilon_2 \in \mathbb{R}$, then there is a norm whose unit disc comprises $\{(\mathbf{v},\mathbf{w}) | ||\mathbf{v}|| \leq \varepsilon_1 \wedge ||\mathbf{w}|| \leq \varepsilon_2\}$ which is equivalent in this sense to the norm described in the previous paragraph.

Let X, Y and Z be Banach spaces and let $\mu : X \times Z \to Y \times Z$ be a a linear transformation from $X \times Z$ to $Y \times Z$. There are linear transformations $\alpha : X \to Y$, $\beta : Z \to Y$, $\gamma : X \to Z$ and $\delta : Z \to Z$, such that such that $\mu = \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix}$ i.e., $(\mathbf{x},\mathbf{z})\mu = (\mathbf{y},\mathbf{w})$ iff:

$$\begin{aligned} \mathbf{y} &= \mathbf{x}\alpha + \mathbf{z}\beta \\ \mathbf{w} &= \mathbf{x}\gamma + \mathbf{z}\delta. \end{aligned}$$

Now $\mathbf{x}\ \underline{\mathsf{trace}\,(\mu)}\ \mathbf{y}$ iff there is a $\mathbf{z}$ such that $(\mathbf{x},\mathbf{z})\mu = (\mathbf{y},\mathbf{z})$, i.e. putting $\mathbf{w} = \mathbf{z}$ in the equations above, iff the following infinite system of equations $(*)$ holds:

$$(*) \qquad \begin{array}{rcl} \mathbf{z} & = & \mathbf{x}\gamma + \mathbf{z}\delta \\ & = & \mathbf{x}\gamma + \mathbf{x}\gamma\delta + \mathbf{z}\delta^2 \\ & = & \mathbf{x}\gamma + \mathbf{x}\gamma\delta + \mathbf{x}\gamma\delta^2 + \mathbf{z}\delta^3 \\ & \cdots & \\ & = & \mathbf{x}\gamma + \mathbf{x}\gamma\delta + \mathbf{x}\gamma\delta^2 + \mathbf{x}\gamma\delta^3 + \ldots + \mathbf{x}\gamma\delta^n + \mathbf{z}\delta^{n+1} \\ & \cdots & \end{array}$$

The trace $\mathsf{trace}\,(\mu)$ will be a total function iff the system of equations $(*)$ has a unique solution for any given $\mathbf{x}$. We are interested in sufficient conditions for this. One algebraic possibility would be to require δ to be nilpotent, i.e., to require $\delta^n = 0$ for some n. However, the idea of taking measurements in a physical system suggests that an analytic criterion would be more appropriate.

Recall that if Z is a normed space, a function $\phi : Z \to Z$ is called a *contraction mapping* if there is $b < 1$ such that for every $\mathbf{z}_1, \mathbf{z}_2 \in Z$, $\mathsf{d}(\mathbf{z}_1\phi, \mathbf{z}_2\phi) < b\mathsf{d}(\mathbf{z}_1,\mathbf{z}_2)$, where d is the metric on Z induced by the norm ($\mathsf{d}(\mathbf{z}_1,\mathbf{z}_2) = ||\mathbf{z}_1 - \mathbf{z}_2||$).

Definition 2.10. Let X, Y and Z be normed spaces:

- The linear transformation $\delta : Z \to Z$ is *strongly contractive* if it is bounded with $||\delta|| < 1$ (equivalently, it is a contraction mapping).
- Let $\alpha : X \to Y$, $\beta : Z \to Y$, $\gamma : X \to Z$ and $\delta : Z \to Z$ be linear transformations. The linear

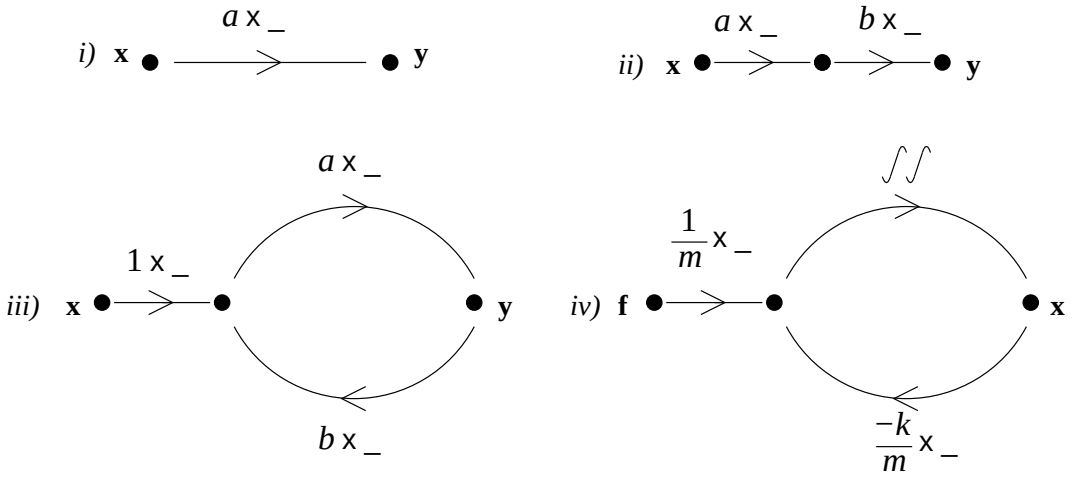


Fig. 2. Some Small Signal Flow Graphs

transformation $\mu = \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix} : X \times Z \to Y \times Z$ is *contractively traced* iff it is bounded (equivalently, each of α, β, γ and δ is bounded) and δ is strongly contractive.

We then have the following proposition, which is related to Banach's contraction mapping theorem for metric spaces.

Proposition 2.11. Let X and Y be normed spaces and let Z be a Banach space. If the linear transformation $\mu : X \times Z \to Y \times Z$ is contractively traced, $\mathsf{trace}\,(\mu)$ is a bounded operator.

Proof. Write $\mu = \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix}$ as in definition 2.10. We have the following estimate for any $n > m$:

$$\begin{aligned} ||\mathbf{x}\gamma\delta^m + \mathbf{x}\gamma\delta^{m+1} + \ldots + \mathbf{x}\gamma\delta^n|| &\leq ||\mathbf{x}\gamma\delta^m|| + ||\mathbf{x}\gamma\delta^{m+1}|| + \ldots + ||\mathbf{x}\gamma\delta^n|| \\ &\leq ||\mathbf{x}|| \cdot ||\gamma|| \cdot (||\delta||^m + ||\delta||^{m+1} + \ldots) \\ &\leq ||\mathbf{x}|| \cdot ||\gamma|| \cdot ||\delta||^m (1 - ||\delta||)^{-1}. \end{aligned}$$

This estimate shows that the series $\Sigma_{n=0}^{\infty} \mathbf{x}\gamma\delta^n$ satisfies the Cauchy criterion and hence is convergent in the Banach space Z. Taking $\mathbf{z} = \Sigma_{n=0}^{\infty} \mathbf{x}\gamma\delta^n$ gives us the desired unique solution to the system of equations $(*)$. We then have that $\mathbf{x}\ \underline{\mathsf{trace}\,(\mu)}\ \mathbf{y}$ iff $\mathbf{y} = \mathbf{x}\alpha + \mathbf{z}\beta$, but $\mathbf{y}$ so-defined is a linear function of $\mathbf{x}$ and we have:

$$||\mathbf{y}|| \leq (||\alpha|| + ||\beta|| \cdot ||\gamma|| \cdot (1 - ||\delta||)^{-1}) ||\mathbf{x}||$$

so $\mathsf{trace}\,(\mu)$ is a bounded operator. □

3. Signal flow graphs and the block diagram language

In this section we discuss notations for specifying linear systems. The signal flow graph notation which we introduce first is familiar from control theory. We view signal flow graphs as akin to unstructured `goto` programs in programming. We then identify an important subclass of signal flow graphs, which we call block diagrams, analogous to structured `while` programs. We then relate the block diagram language with the approach based on traced monoidal categories of [AMMO09].

3.1. Signal flow graphs

A signal flow graph is built upon a directed graph G comprising a set N of nodes and a set E of edges each edge e having a source node s_e and a target node t_e. For each node n we have a vector space V_n and for

each edge e we have a linear transformation $f_e : V_{s_e} \to V_{t_e}$. We view an edge e in a signal flow graph as passing a value assigned to its source node on to its target node subject to the linear transformation f_e. Two nodes I and O are distinguished as the input and output nodes. A *flow* assigns to each node n a value in the vector space V_n in such a way that the value assigned to each node is equal to the sum of the values passed in. The set of all flows defines an input-output relation r_G between the vector spaces labelling the input and output nodes. See [AMMO07] for a more detailed description. For example, consider the input-output relations of the four signal flow graphs shown in Figure 2. In these diagrams, the linear transformation are scalar multiplication with the exception of the double integrator in *iv)*. In each diagram, I comprises the left-most node and O comprises the right-most node. The input-output relations of these signal flow graphs are then as follows:

(i) $\{(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} = a\mathbf{x}\}$
(ii) $\{(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} = ab\mathbf{x}\}$
(iii) $\{(\mathbf{x}, \mathbf{y}) \mid (1 - ab)\mathbf{y} = a\mathbf{x}\}$
(iv) $\{(f, x) \mid m\ddot{x}(t) + kx(t) = f\}$ (the system of Figure 1).

(for *(iii)*, note that $\mathbf{y} = a(\mathbf{x} + b\mathbf{y})$ and rearrange).

A flow on G can be viewed as an element of the cartesian product $\prod_{n \in N} V_n$ and the set of all flows on G is non-empty and is closed under addition and scalar multiplication, i.e., the flows comprise a subspace of $\prod_{n \in N} V_n$. The relation r_G is then the image of this subspace under the projection of $\prod_{n \in N} V_n$ onto $V_I \times V_O$. Since this projection is a linear transformation, its image r_G is a subspace of $V_I \times V_O$, i.e., it is an additive relation.

3.2. Block diagram language – In practice

We now look at a subclass of signal flow graphs that are built up recursively from simple building blocks using standard constructions. These are the standard block diagram constructors used in practice, in languages such as Simulink, for instance. In the following Section 3.3 we describe a refinement of this language (the two in fact turn out to be equivalent) which will be used to proof the soundness theorem.

Let $\mathcal{N}$ be a set of names and let $\mathcal{T}$ be the set of syntactic sorts obtained by drawing basic sorts from $\mathcal{N}$ and forming new sorts from old using the cartesian product operator, $\times$. Let Σ be a many-sorted signature providing at least the following, where V and the V_i range over $\mathcal{T}$:

- Scalar multiplication operators: $q \times \_ : V \to V$, for $q \in \mathbb{Q}$.
- Projection operators: $\pi_i : V_1 \times \ldots V_n \to V_i$, for $1 \leq i \leq n \in \mathbb{N}$.
- Injection operators: $\iota_i : V_i \to V_1 \times \ldots V_n$

In addition, Σ, may contain a set Φ of other operator symbols, $\phi_i : V_i \to W_i$ for selected $V_i, W_i \in \mathcal{T}$. We assume given a structure $\mathcal{S}$ for Σ in which each name in $\mathcal{N}$ is interpreted as a Banach space, in which $\times$, $q \times \_$, π_i and ι_i have their usual interpretations and in which the ϕ_i are interpreted as bounded operators. For example, if $\mathcal{N} = \{V\}$, $\Phi = \emptyset$ and we interpret V as $\mathbb{R}$, the structure is the subcategory of the category of all Banach spaces whose objects are the spaces $\mathbb{R}^n$, $n \in \mathbb{N}$ and whose morphisms comprise the linear transformations whose matrix expressions with respect to the standard bases have rational coefficients.

Definition 3.1. The language $BD_{\mathcal{S}}$ of *block diagrams over* $\mathcal{S}$ is a typed language defined as follows with semantics in the category of vector spaces and additive relations:

(i) If f is a term over the signature Σ, then f is a block diagram of type $V \to W$ where V and W name the domain and range of f. (Semantics: $[\![f]\!]$ denotes the bounded operator interpreting f in $\mathcal{S}$)
(ii) If B is a block diagram of type $U \to V$ and C is a block diagram of type $V \to W$, then $B; C$ is a block diagram of type $U \to W$. (Semantics: sequential composition: $[\![B; C]\!] = ([\![B]\!]; [\![C]\!])$).
(iii) If B is a block diagram of type $V \to W$ and C is a block diagram of type $V \to W$, then $\mathsf{Sum}(B, C)$ is a block diagram of type $V \to W$. (Semantics: pointwise sum: $[\![\mathsf{Sum}(B, C)]\!] = [\![B]\!] + [\![C]\!]$).
(iv) If B is a block diagram of type $V \to W$ and C is a block diagram of type $W \to V$, then $\mathsf{Loop}(B, C)$ is a block diagram of type $V \to W$. (Semantics: feedback look: $[\![\mathsf{Loop}(B, C)]\!] = \mathsf{loop}\,([\![B]\!], [\![C]\!])$).

There are many signal flow graphs which are not block diagrams in this sense, but it is proved in [AMMO07] that given an adequate supply of primitive linear transformations, the input-output relation of any signal flow graph is also the input-output relation of a block diagram. This is analogous to the Turing-completeness of `while` programs: our structured block diagrams are as expressive as the unstructured signal flow graphs.

Throughout the sequel we work in the subcategory $\mathcal{C} = \mathcal{C}_{\mathcal{S}}$ of the category $\mathcal{B}_+$ of Banach spaces and additive relations generated by the structure $\mathcal{S}$ that parametrises the block diagram language of interest.

3.3. Block diagram language – In theory

As mentioned above, for the soundness theorem it will be convenient to give the semantic domains of our language the structure of a traced monoidal category. We will then be able to invoke the general soundness theorem from the recent work on abstract Hoare logic [AMMO09].

Let us assume that amongst the basic linear transformations of the given structure $\mathcal{S}$ we have summation, splitting and twist, which implies we have corresponding named block diagrams:

- Summation node. $(+) : V \times V \to V$ (Semantics: addition function: $[\![(+)(\mathbf{v}, \mathbf{w})]\!] = [\![\mathbf{v}]\!] + [\![\mathbf{w}]\!]$).
- Splitting node. $\nabla : V \to V \times V$ (Semantics: diagonal function: $[\![\nabla(\mathbf{v})]\!] = ([\![\mathbf{v}]\!], [\![\mathbf{v}]\!])$).
- Twist. $\mathsf{s} : V \times W \to W \times V$ (Semantics: interchange function: $[\![\mathsf{s}(\mathbf{v}, \mathbf{w})]\!] = ([\![\mathbf{w}]\!], [\![\mathbf{v}]\!])$).

Instead of taking $\mathsf{Sum}(B, C)$ and $\mathsf{Loop}(B, C)$ as block diagram constructors (as in Section 3.2) we could alternatively consider the following as the basic block diagram constructors

- If B is a block diagram of type $U \to V$ and C is a block diagram of type $W \to Z$, then $\mathsf{Par}(B, C)$ is a block diagram of type $U \times W \to V \times Z$. (Semantics: parallel composition: $[\![\mathsf{Par}(B, C)]\!] = [\![B]\!] \times [\![C]\!]$).
- If B is a block diagram of type $V \times Z \to W \times Z$, then $\mathsf{Trace}\,(B)$ is a block diagram of type $V \to W$ (Semantics: trace: $[\![\mathsf{Trace}\,(B)]\!] = \mathsf{trace}\,([\![B]\!])$).

Using Theorem 2.9 it is easy to see that the constructors of Definition 3.1 are definable in terms of sequential composition and the parallel composition and trace constructors. In the following section we derive a Hoare logic for block diagram systems built via sequential composition, parallel composition and trace. We write $\mathcal{L}^b$ for the set of the basic systems, which we assume to include summation, splitting and twist.

4. Hoare Logic

We come now to the main contribution of the paper, which is the sound Hoare logic system shown in Figure 3. As we formally define below, our assertions b_0 and b_1 are closed balls in the input and output vector space on which A operates (more about the assertion language in Section 4.4). Hence, a Hoare triple such as $\{b_0\}A\{b_1\}$ should be read as: If the input signal belongs to the closed ball b_0 then the output of the system A on that input signal will output a signal in the closed ball b_1. A system of Hoare logic rules like the ones in Figure 3 are called *sound* if whenever the Hoare triples in the premise of the rule are true (according the to the reading above) so is the Hoare triple in the conclusion. Moreover, the proof system is called *complete* if all true Hoare triples can be proved in the system.

In order to prove that these rules are sound for reasoning about systems described via the block diagram language BD of Sections 3.2 and 3.3, we will make use of the machinery of abstract Hoare logic, recently developed in [AMMO09]. Reader already who are only interested in the practical applications of these proof rules may skip directly to Sections 4.4 and 5.

Recall from [AMMO09] that the semantic interpretation of the Hoare triples assume a given structure $\mathcal{S}$ giving the block-diagram language a semantics. In our case, the structure $\mathcal{S}$, will be in the category of Banach spaces and bounded linear transformations, and the actual denotations of block diagrams in a subcategory $\mathcal{C} = \mathcal{C}_{\mathcal{S}}$ of the category of Banach spaces and additive relations. We will first recap the notions of an abstract Hoare logic and verification functor. Then we present a concrete sound verification functor $H : \mathcal{C} \to \mathcal{H}$ giving rise to a sound Hoare logic for reasoning about systems whose denotations are in $\mathcal{C}$.

$$\frac{}{\{b_0\}A\{b_1\}}\,(\mathsf{Ax})\;(\dagger) \qquad \frac{\{b_0\}A\{b_1\}}{\{b_0'\}A\{b_1'\}}\,(\mathsf{Con})\;(\ddagger)$$

$$\frac{\{b_0\}A\{b_2\} \quad \{b_1\}B\{b_3\}}{\{\langle b_0, b_1\rangle\}\mathsf{Par}(A,B)\{\langle b_2, b_3\rangle\}}\,(\mathsf{Par})$$

$$\frac{\{b_0\}A\{b_1\} \quad \{b_1\}B\{b_2\}}{\{b_0\}A;B\{b_2\}}\,(\mathsf{Seq}) \qquad \frac{\{\langle b_0, b_2\rangle\}A\{\langle b_1, b_2\rangle\}}{\{b_0\}\mathsf{Trace}\,(A)\{b_1\}}\,(\mathsf{Fb}_+)$$

Fig. 3. The system $\mathbf{HL}(\mathcal{L}, [\![\cdot]\!], H)$ with $[\![\cdot]\!] : \mathcal{L} \to \mathcal{S}$ and $H : \mathcal{S} \to \mathcal{H}$.

4.1. Abstract Hoare logic

In this section we give a short summary of the abstract Hoare logic theory developed in [AMMO09]. A relation $r : X \leftrightarrow Y$ between two pre-ordered sets $(X, \sqsubseteq_X)$ and $(Y, \sqsubseteq_Y)$ is called *monotone*

- if $P \mathrel{\underline{r}} Q$ and $P' \sqsubseteq_X P$ and $Q \sqsubseteq_Y Q'$ imply $P' \mathrel{\underline{r}} Q'$.

Let $\mathcal{H}$ denote the category of pre-ordered sets and monotone relations. We call $\mathcal{H}$ the *Hoare category*. It is easy to check that $\mathcal{H}$ is a symmetric traced monoidal cateogry, with cartesian product as the monoidal operator and the standard trace for cartesian products. The identity of composition in the category $\mathcal{H}$ is $1_X :\equiv \sqsubseteq_X$, which is monotone by the transitivity of $\sqsubseteq_X$.

Definition 4.1 (Verification functor, [MMO06, AMMO09]). For any traced monoidal category $\mathcal{S}$, a functor $H : \mathcal{S} \to \mathcal{H}$ is called a *sound and complete verification functor* if H is a traced monoidal functor. If H only maps the structure of $\mathcal{S}$ inside that of $\mathcal{H}$ as

$$\begin{aligned} \mathsf{trace}\,(H(f)) &\subseteq H(\mathsf{Trace}\,(f)) \\ H(f); H(g) &\subseteq H(f; g) \\ H(f) \times H(g) &\subseteq H(f \times g) \end{aligned}$$

then it is called a *sound verification functor*.

Any verification functor $H : \mathcal{S} \to \mathcal{H}$ gives rise to a Hoare logic as follows. Let A be a concrete block diagram, whose meaning is a morphism $[\![A]\!] : X \to Y$ in $\mathcal{S}$, and let $P \in H(X)$ and $Q \in H(Y)$. Define *abstract Hoare triples* semantically as

$$\{P\}A\{Q\} \;:\equiv\; P \mathrel{\underline{H([\![A]\!])}} Q. \tag{2}$$

P and Q here are the semantic values of what, in a practice, will be syntactic assertions about the inputs and outputs of the system (see Section 4.4). The main theorem of [AMMO09] is that:

Theorem 4.2 ([MMO06, AMMO09]). If H is a sound (and complete) verification functor then the derived Hoare logic system (shown in Figure 3) is sound (and complete).

4.2. The verification functor $H : \mathcal{C} \to \mathcal{H}$

Our goal now is to define a concrete *sound* verification functor $H : \mathcal{C} \to \mathcal{H}$. For notational simplicity, we assume that the structure $\mathcal{S}$ defining our category $\mathcal{C}$ includes a single named vector space V so that the objects of $\mathcal{C}$ are the finite products V^n. The extension to the general case is straightforward.

Let $B(\mathbf{v}, \delta)$ denote the closed ball in the space V with centre $\mathbf{v}$ and radius δ. We define H as follows: H maps the vector space V to the pre-ordered set of closed V-balls, i.e.

$$H(V) :\equiv \{B(\mathbf{v}, \delta) \;:\; \mathbf{v} \in V \wedge \delta \in \mathbb{R}^+\},$$

where $B(\mathbf{v},\delta) \sqsubseteq B(\mathbf{w},\eta)$ if the closed ball $B(\mathbf{v},\delta)$ is contained in the closed ball $B(\mathbf{w},\eta)$. On V^n, H is defined as

$$H(V^n) :\equiv (H(V))^n,$$

where the pre-order on the tuples of $H(V^n)$ is the pointwise pre-order. So, H maps objects of $\mathcal{C}$ to pre-ordered sets in $\mathcal{H}$. Let us refer to tuples of balls as *boxes*. From now on we will use variables $b, b_0, b_1, \ldots$ to range over boxes.

On the morphisms of $\mathcal{C}$, H extends a given additive relation $r : V^n \leftrightarrow V^m$ to a monotone relation $H(r) : H(V^n) \leftrightarrow H(V^m)$ as follows:

$$b_0 \,\underline{H(r)}\, b_1 \;:\equiv\; \forall \mathbf{v} \in b_0 \forall \mathbf{w}(\mathbf{v}\,\underline{r}\,\mathbf{w} \to \mathbf{w} \in b_1).$$

Intuitively, $b_0 \,\underline{H(r)}\, b_1$ holds whenever the relational image of r on the box b_0 is contained in the box b_1. If we define the "strongest post-condition" of a relation r as[1]

$$\mathsf{sp}(r, b_0) :\equiv \{\mathbf{w} \;:\; \exists \mathbf{v} \in b_0(\mathbf{v}\,\underline{r}\,\mathbf{w})\}$$

then we can also write the relation $H(r)$ as

$$b_0 \,\underline{H(r)}\, b_1 \;:\equiv\; \mathsf{sp}(r, b_0) \subseteq b_1.$$

Hence in our derived Hoare logic, Hoare triples $\{b_0\}A\{b_1\}$ will denote $\mathsf{sp}([\![A]\!], b_0) \subseteq b_1$. It is easy to check that $H(r)$ is a monotone relation.

Next we show that H is indeed a sound verification functor. We will require, however, an extra condition in order to obtain soundness of the feedback rule. Let us denote by $\langle b_0, b_1\rangle \in H(X \times Y)$ $(= H(X) \times H(Y))$ the pairing of two balls $b_0 \in H(X)$ and $b_1 \in H(Y)$.

Definition 4.3. A block diagram $A\colon X \times Z \leftrightarrow Y \times Z$ is called *valid* if it satisfies

$$(+)\;\; \{\langle b_0, b_2\rangle\}A\{\langle b_1, b_2\rangle\} \to \forall \mathbf{x} \in b_0 \forall \mathbf{z}(\exists \mathbf{y}(\langle \mathbf{x}, \mathbf{z}\rangle \,\underline{[\![A]\!]}\, \langle \mathbf{y}, \mathbf{z}\rangle) \to \mathbf{z} \in b_2).$$

Intuitively, condition $(+)$ says that a fixed point set b_2 must contain all actual fixed points (for parameters in b_0). This condition holds, for instance, when $[\![A]\!]$ is a "contractive relation", as the following lemma shows.

Lemma 4.4. If $[\![A]\!]$ is a contractively traced linear transformation (see Definition 2.10) then A satisfies condition $(+)$.

Proof. As $[\![A]\!]$ is a linear transformation, we can write it in block matrix form:

$$[\![A]\!] = \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix}$$

where, as $[\![A]\!]$ is contractively traced, δ is strongly contractive. If $\{\langle b_0, b_2\rangle\}A\{\langle b_1, b_2\rangle\}$ holds, then, in particular, we have that $(*)$ $\mathsf{sp}(\gamma, b_0) + \mathsf{sp}(\delta, b_2) \subseteq b_2$. Fix $\mathbf{x} \in b_0$. Since δ is a contractive mapping, the mapping $\mathbf{z} \mapsto \mathbf{x}\gamma + \mathbf{z}\delta$ is a contraction mapping, which, by $(*)$ maps b_2 to b_2. By Banach's fixed point theorem, it has a unique fixed point in b_2 and this satisfies $\mathbf{z} = \mathbf{x}\gamma + \mathbf{z}\delta$ (cf. Proposition 2.11). □

Lemma 4.5. The following properties hold of $\mathsf{sp}([\![A]\!], b)$

$$\begin{array}{lll}
(i) & \exists b_1(\mathsf{sp}([\![A]\!], b_0) \subseteq b_1 \wedge \mathsf{sp}([\![B]\!], b_1) \subseteq b_2) & \Rightarrow \quad \mathsf{sp}([\![A]\!]; [\![B]\!], b_0) \subseteq b_2 \\
(ii) & \mathsf{sp}([\![A]\!], b_0) \subseteq b_2 \wedge \mathsf{sp}([\![B]\!], b_1) \subseteq b_3 & \Rightarrow \quad \mathsf{sp}([\![A]\!] \times [\![B]\!], \langle b_0, b_1\rangle) \subseteq \langle b_2, b_3\rangle \\
(iii) & \exists b_2(\mathsf{sp}([\![A]\!], \langle b_0, b_2\rangle) \subseteq \langle b_1, b_2\rangle) & \Rightarrow \quad \mathsf{sp}(\mathsf{trace}\,([\![A]\!]), b_0) \subseteq b_1,
\end{array}$$

where for (iii) we assume that A satisfies $(+)$.

Proof. The only non-trivial implication is (iii). Assume $\mathsf{sp}([\![A]\!], \langle b_0, b_2\rangle) \subseteq \langle b_1, b_2\rangle$, for some b_2. By condition $(+)$, for each $\mathbf{v} \in b_0$ all fixed points $\mathbf{w}$ are in b_2. Hence, $\langle \mathbf{v}, \mathbf{w}\rangle \,\underline{r}\, \langle \mathbf{v}', \mathbf{w}\rangle$ implies $\mathbf{v}' \in b_1$. □

[1] Note that the strongest post-condition might not be a box. For instance, the strongest post-condition of split is $\mathsf{sp}(\nabla, b_0) \equiv \{\langle \mathbf{v}, \mathbf{v}\rangle \;:\; \mathbf{v} \in b_0\}$, which is not a box.

Proposition 4.6. H defined above is a sound monoidal functor.

Proof. It is clear that $H(1_{V^n}) = 1_{H(V^n)} = \sqsubseteq_{H(V^n)}$. Let us look at composition. Using Lemma 4.5 (i), we have

$$\begin{array}{rcl} b_0\, \underline{H(s); H(r)}\, b_2 & \equiv & \exists b_1 (b_0\, \underline{H(s)}\, b_1 \wedge b_1\, \underline{H(s)}\, b_2) \\ & \equiv & \exists b_1 (\mathsf{sp}(r, b_0) \subseteq b_1 \wedge \mathsf{sp}(s, b_1) \subseteq b_2) \\ & \Rightarrow & \mathsf{sp}(r; s, b_0) \subseteq b_2 \\ & \equiv & b_0\, \underline{H(r; s)}\, b_2. \end{array}$$

Note that elements of $H(V^n)$ are n-tuples of balls. That H maps soundly the monoidal structure of $\mathcal{C}$ can be seen, using Lemma 4.5 (ii), as follows:

$$\begin{array}{rcl} \langle b_0, b_1\rangle\, \underline{H(r) \times H(s)} \langle b_2, b_3\rangle & \equiv & b_0\, \underline{H(s)}\, b_2 \wedge b_1\, \underline{H(s)}\, b_3 \\ & \equiv & \mathsf{sp}(r, b_0) \subseteq b_2 \wedge \mathsf{sp}(s, b_1) \subseteq b_3 \\ & \Rightarrow & \mathsf{sp}(r \times s, \langle b_0, b_1\rangle) \subseteq \langle b_2, b_3\rangle \\ & \equiv & \langle b_0, b_1\rangle\, \underline{H(r \times s)} \langle b_2, b_3\rangle. \end{array}$$

That concludes the proof. □

Proposition 4.7. H defined above is a sound verification functor, for systems satisfying (+).

Proof. It remains to be checked that H maps the trace structure of $\mathcal{C}$ into the trace structure of $\mathcal{H}$. Assuming r satisfies (+), by Lemma 4.5 (iii) we have

$$\begin{array}{rcl} b_0\, \underline{\mathsf{trace}\,(H(r))}\, b_1 & \equiv & \exists b_2 (\langle b_0, b_2\rangle\, \underline{H(r)} \langle b_1, b_2\rangle) \\ & \equiv & \exists b_2 (\mathsf{sp}(r, \langle b_0, b_2\rangle) \subseteq \langle b_1, b_2\rangle) \\ & \Rightarrow & \mathsf{sp}(\mathsf{trace}\,(r)\,, b_0) \subseteq b_1 \\ & \equiv & b_0\, \underline{H(\mathsf{trace}\,(r)))}\, b_1. \end{array}$$

That concludes the proof. □

Let us say a system is *contractively traced* iff for every subsystem of the form $\mathsf{Trace}\,(A)$, the bounded linear transformation $[\![A]\!]$ is contractively traced.

Proposition 4.8. H defined above is a sound verification functor for contractively traced systems. Moreover every such system denotes a total bounded operator.

Proof. This is immediate from Proposition 4.7, Lemma 4.4 and Proposition 2.11. □

4.3. The derived Hoare logic

We will denote by $\mathbf{HL}(\mathcal{L}, [\![\cdot]\!], H)$ the set of rules shown in Figure 3. The side conditions for the rule in Figure 3 are:

(†) $A \in \mathcal{L}^b$ and $b_0\, \underline{H([\![A]\!])}\, b_1$ (i.e. $\mathsf{sp}([\![A]\!], b_0) \subseteq b_1$)

(‡) $b_0' \sqsubseteq b_0$ and $b_1 \sqsubseteq b_1'$.

Hence, our Hoare logic is developed relative to a decision procedure used for checking simple Hoare triples, i.e. Hoare triples $\{b_0\}A\{b_1\}$, for $A \in \mathcal{L}^b$.

Theorem 4.9 (Soundness Theorem). The rules given in Figure 3 are sound.

Proof. Follows from the main result of [AMMO09] (cf. Theorem 4.2) and Proposition 4.7. The soundness of the feedback rules uses the assumption that the denotation of A has property (+). □

In practise, instead of using condition (+) we will rather use a less general rule for the feedback which has the advantage that (a more restrictive form of) the corresponding side condition can be expressed in the assertion language, namely, for $0 < \varepsilon < 1$,

$$\frac{\{\langle b_0, \mathbf{v}_1 \pm \delta_1, \ldots, \mathbf{v}_n \pm \delta_n\rangle\} A \{\langle b_1, \mathbf{v}_1 \pm \varepsilon\delta_1, \ldots, \mathbf{v}_n \pm \varepsilon\delta_n\rangle\}}{\{b_0\}\mathsf{Trace}\,(A)\{b_1\}}\ (\mathsf{Fb}_\varepsilon)$$

Intuitively, the rule Fb_ε ensures that A is "contractive" on the second argument, which implies that fixed points exists and are unique in the fixed point box b_2. This version of the feedback rule corresponds to the Hoare logic rule for total correctness of a while loop, in the sense that any system proved correct via this more restrictive rule will be guaranteed to be functional (i.e., the output exists and is unique for any input satisfying the pre-condition).

Proposition 4.10. If $\{\langle b_0, \mathbf{v}_1 \pm \delta_1, \ldots, \mathbf{v}_n \pm \delta_n\rangle\} A \{\langle b_1, \mathbf{v}_1 \pm \varepsilon\delta_1, \ldots, \mathbf{v}_n \pm \varepsilon\delta_n\rangle\}$ holds for some $\varepsilon, b_0, b_1, \mathbf{v}_i$'s and δ_i's then $[\![A]\!]$ is contractively traced (cf. Definition 2.10).

Proof. Our assumption $\{\langle b_0, \mathbf{v}_1 \pm \delta_1, \ldots, \mathbf{v}_n \pm \delta_n\rangle\} A \{\langle b_1, \mathbf{v}_1 \pm \varepsilon\delta_1, \ldots, \mathbf{v}_n \pm \varepsilon\delta_n\rangle\}$ can be written as

$$\mathsf{sp}([\![A]\!], \langle b_0, \mathbf{v}_1 \pm \delta_1, \ldots, \mathbf{v}_n \pm \delta_n\rangle) \subseteq \langle b_1, \mathbf{v}_1 \pm \varepsilon\delta_1, \ldots, \mathbf{v}_n \pm \varepsilon\delta_n\rangle.$$

Assuming

$$[\![A]\!] = \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix},$$

we obtain $\mathsf{sp}(\beta, b_0) + \mathsf{sp}(\delta, \mathbf{v}_1 \pm \delta_1, \ldots, \mathbf{v}_n \pm \delta_n) \subseteq \mathbf{v}_1 \pm \varepsilon\delta_1, \ldots, \mathbf{v}_n \pm \varepsilon\delta_n$. Let $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ be the mid point of the box b_0. Because β and δ are linear transformations we have

$$\mathsf{sp}(\beta, b_0 - \mathbf{v}) + \mathsf{sp}(\delta, \mathbf{v}_1 \pm \delta_1, \ldots, \mathbf{v}_n \pm \delta_n) \subseteq (\mathbf{v}_1 \pm \varepsilon\delta_1, \ldots, \mathbf{v}_n \pm \varepsilon\delta_n) - (\mathbf{w}_1, \ldots, \mathbf{w}_k)\beta,$$

where the box $b_0 - (\mathbf{w}_1, \ldots, \mathbf{w}_k)$ is centred at the origin. This implies that $\mathsf{sp}(\beta, b_0 - (\mathbf{w}_1, \ldots, \mathbf{w}_k))$ also contains the origin, and hence we can conclude

$$\mathsf{sp}(\delta, \mathbf{v}_1 \pm \delta_1, \ldots, \mathbf{v}_n \pm \delta_n) \subseteq (\mathbf{v}_1 \pm \varepsilon\delta_1, \ldots, \mathbf{v}_n \pm \varepsilon\delta_n) - (\mathbf{w}_1, \ldots, \mathbf{w}_k)\beta,$$

which means that δ is strongly contractive with $||\delta|| \leq \varepsilon$ with respect to the norm whose unit ball is the box $\mathbf{v}_1 \pm \delta_1, \ldots, \mathbf{v}_n \pm \delta_n$. □

Let $\mathbf{HL}_0(\mathcal{L}, [\![\cdot]\!], H)$ be the set of rules obtained from those shown in Figure 3 by replacing Fb_+ by Fb_ε, $0 < \varepsilon < 1$. Our final theorem in this section shows that this set of rules give a syntactic characterization of a very tractable subset of the block diagram language.

Theorem 4.11. The rules $\mathbf{HL}_0(\mathcal{L}, [\![\cdot]\!], H)$ are sound. Moreover, if A is any system appearing in a proof tree whose leaves are all instances of the rules Ax, then $[\![A]\!]$ is a total bounded linear operator.

Proof. This is immediate from Theorem 4.9 and Propositions 4.8 and 4.10. □

4.4. The assertion language

In the Hoare logic developed above we have used "semantical" boxes as pre- and post-conditions, in the sense that we have not specified a syntactic assertion language to describe boxes. In this section we briefly suggest an appropriate language for describing boxes, and analyse the kinds of verification conditions which are generated during a proof, and how these could possibly be proved automatically (via a theorem prover).

Note that a box is a sequence of closed balls $B(\mathbf{v}, \varepsilon)$ where $\mathbf{v}$ is a vector in V, and ε is a positive rational number $\mathbb{Q}_+^*$. Therefore, we consider two term languages $\mathcal{L}_V$ and $\mathcal{L}_\varepsilon$, one for describing vectors in V and the other for describing the "error" ε. For instance, consider the Hoare triple

$$\{\langle \mathbf{v} \pm \varepsilon, \mathbf{w} \pm \delta\rangle\}(+)\{(\mathbf{v} + \mathbf{w}) \pm (\varepsilon + \delta)\}$$

which says that by adding two vectors belonging to the balls $B(\mathbf{v}, \varepsilon)$ and $B(\mathbf{w}, \delta)$ we will obtain a vector which belongs to the ball centered at $\mathbf{v} + \mathbf{w}$ with a radius of $\varepsilon + \delta$. So, we would want our assertion language to contain primitives summing operators for both vectors (in $\mathcal{L}_V$) and error margins (in $\mathcal{L}_\varepsilon$).

Formally, the language $\mathcal{L}_V$ contains a countable number of variables $\mathbf{v}, \mathbf{w}, \ldots$, a constant zero vector (0), and vector operations such as addition $(\mathbf{v} + \mathbf{w})$, scalar multiplication ($a \cdot \mathbf{v}$, with $a \in \mathbb{R}^+_*$), and other basic operations corresponding to the linear transformations of the basic building blocks (e.g. integration or differentiation). The language $\mathcal{L}_\varepsilon$ contains variables $\varepsilon, \delta, \ldots$ ranging over positive rational numbers, all positive rational numbers as constants, and the standard operations on rational such as addition, multiplication and division[2].

Given a closed vector term $t \in \mathcal{L}_V$ and a closed positive rational term $r \in \mathcal{L}_\varepsilon$ we write $t \pm r$ for the expression denoting a ball $B(t, r)$ in V. If t and r have free variables, then $t \pm r$ can be seen as a mapping from its free-variables to balls in V. Checking a verification condition $t \pm r \subseteq t' \pm r'$ amounts to checking the truth of the expression $|t - t'| \leq r' - r$.

One possible use of the Hoare logic described above is as follows. Given a block diagram A, we first encode it in the refined block diagram language described in Section 3.3. This gives us an equivalent block diagram A'. Let b_0 be an abstract box, in the sense that all its balls are of the form $\mathbf{v} \pm \varepsilon$, here $\mathbf{v}$ and ε are variables in $\mathcal{L}_V$ and $\mathcal{L}_\varepsilon$, respectively. We can use the Hoare logic rules to search for a proof of $\{b_0\}A\{b_1\}$, for some box expression b_1, over the free-variables of b_0 plus extra free-variables introduced in the trace rule. Also, each trace rule used in the proof will generate a verification condition of the form $t \pm r \subseteq t' \pm r'$. Let us call $\vec{x}$ the tuple of all free-variables in the final proof, and $\mathsf{VC}(\vec{x})$ the sequence of verification conditions generated. What we then obtain is the following implication

$$\forall \vec{x}(\mathsf{VC}(\vec{x}) \to \{b_0\}A'\{b_1\}).$$

At this point we can ask a theorem prover to solve the inequalities $\mathsf{VC}(\vec{x})$, and given a solution $\vec{e}$, we finally obtain a proof of $\{b_0[\vec{e}/\vec{x}]\}A'\{b_1[\vec{e}/\vec{x}]\}$, which implies $\{b_0[\vec{e}/\vec{x}]\}A\{b_1[\vec{e}/\vec{x}]\}$.

5. Applications

In this section we give examples that illustrate our proof rules in the important special case of linear transformations between vector spaces over some field K. We think of subsets, A, of a vector space, V, as assertions about elements of V. For example, in the x-y plane, the x-axis is thought of as the assertion $y = 0$. If $r : V \to W$ is a linear transformation, and A and B are subsets of V, W respectively, we use the *Hoare triple* notation $\{A\}\, r\, \{B\}$ to mean $Ar \subseteq B$ (or $rA \subseteq B$, if you prefer functions on the left), i.e., if A holds of the input of r, then B holds of its output.

5.1. Simple example

We will now apply our Hoare logic to simple dynamic systems of the sort discussed in section 1. For simplicity, we consider evoluation of systems over a unit time interval. So, for example, the system of Figure 1 is given by $\frac{1}{m}; \mathsf{Trace}\big((+); \int; \int; (1 \times \frac{-k}{m})\big)$, which we interpret in the Banach space $\mathcal{L}_1[0, 1]$ (see Section 2.3, we now take $t = 1$). We first note that the following Hoare triples (for basic systems) are derivable:

(a) $\{\langle \mathbf{v} \pm \varepsilon, \mathbf{w} \pm \delta \rangle\}(+)\{(\mathbf{v} + \mathbf{w}) \pm (\varepsilon + \delta)\}$

(b) $\big\{\mathbf{v} \pm \varepsilon\big\} \int \big\{(\int \mathbf{v}) \pm \varepsilon\big\}$

(c) $\big\{\mathbf{v} \pm \varepsilon\big\} \int \int \big\{(\int \int \mathbf{v}) \pm \varepsilon/2\big\}$.

Let us illustrate the approach outlined in Section 4.4. We starting by picking a parameter box $\mathbf{v} \pm \varepsilon$, which we assume contains the input to our system. We then use the Hoare logic rules to derive a post-condition box $b := \frac{\int \int \mathbf{v} + m\mathbf{w}}{m} \pm \frac{\varepsilon + m\delta}{2m}$ for the given pre-condition $\mathbf{v} \pm \varepsilon$ as

[2] One could also think of saying that for each expression $\mathbf{v}$ in $\mathcal{L}_V$, the norm of $\mathbf{v}$, i.e. $||\mathbf{v}||$, is an expression in $\mathcal{L}_\varepsilon$. Since $||\mathbf{v}||$ might not be a positive rational number, we leave that out for the moment.

$$\dfrac{\dfrac{\left\{(\dfrac{\mathbf{v}+m\mathbf{w}}{m})\pm(\dfrac{\varepsilon+m\delta}{m})\right\}\int\int\left\{b\right\}\quad\left\{b\right\}\dfrac{-k}{m}\left\{(\dfrac{-k\int\int\mathbf{v}+m\mathbf{w}}{m^2})\pm\dfrac{-k(\varepsilon+m\delta)}{2m^2}\right\}}{\left\{\langle\dfrac{\mathbf{v}}{m}\pm\dfrac{\varepsilon}{m},\mathbf{w}\pm\delta\rangle\right\}(+);\int;\int;(1\times\dfrac{-k}{m})\left\{\langle b,(\dfrac{-k\int\int\mathbf{v}+m\mathbf{w}}{m^2})\pm\dfrac{-k(\varepsilon+m\delta)}{2m^2}\rangle\right\}}\,(\mathsf{Fb})}{\dfrac{\left\{\dfrac{\mathbf{v}}{m}\pm\dfrac{\varepsilon}{m}\right\}\mathsf{Trace}\left((+);\int;\int;(1\times\dfrac{-k}{m})\right)\left\{b\right\}}{\left\{\mathbf{v}\pm\varepsilon\right\}\dfrac{1}{m};\mathsf{Trace}\left((+);\int;\int;(1\times\dfrac{-k}{m})\right)\left\{b\right\}}\,(\mathsf{Seq})}\,(\mathsf{Seq})$$

This derivation has a verification condition (VC) $\frac{-k\int\int\mathbf{v}+m\mathbf{w}}{m^2}\pm\frac{-k(\varepsilon+m\delta)}{2m^2}\subseteq\mathbf{w}\pm\delta$ triggered by the feedback rule. Hence, what we effectively derived is the implication

$$(\text{VC})\ \Rightarrow\ \left\{\mathbf{v}\pm\varepsilon\right\}\frac{1}{m};\mathsf{Trace}\left((+);\int;\int;(1\times\frac{-k}{m})\right)\left\{b\right\}$$

giving an estimate of the position of the cart given an estimate of the force applied. Under any pre-condition $\mathbf{v}\pm\varepsilon$ satisfying (VC) we have that not only the above system is functional (cf. discussion after Theorem 4.9) but it ensures that the position of the cart is somewhere inside the ball b.

5.2. Linear Filters

For another example, we consider the specification of linear filters. Our linear filters operate on signals which are conveniently represented as complex-valued functions of a real variable: we take the space S of *signals* to be the Banach space $\mathcal{L}^{\mathbb{C}}_2[-\pi,\pi]$ of complex-valued functions f on the interval $[-\pi,\pi]$ for which the Lebesgue integral $\int_{[-\pi,\pi]}(f\overline{f})$ exists under the norm given by that integral. Any signal has a unique expression as a Fourier series $f(\theta)=\sum_{\omega=1}^{\infty}a_\omega e^{\mathsf{i}\omega\theta}$ where each component $a_\omega e^{\mathsf{i}\omega\theta}=a_\omega(\mathsf{cos}\theta+\mathsf{i}\,\mathsf{sin}\theta)$ corresponds to a pure sinusoidal signal of frequency ω and amplitude a_ω (with respect to some suitable choice of units). In other words, writing $\mathbf{e}_\omega$ for the functions $\theta\mapsto e^{\mathsf{i}\omega\theta}$, the $\mathbf{e}_\omega$ form what is called a *Hilbert basis* for S.

A linear filter is a linear transformation $g:S\to S$ of a type that can be implemented by a digital system that samples a physical signal at regular intervals and outputs a weighted average of the sample values calculated over some window of sample points. It can be shown that the effect of such a digital filter is to multiply each amplitude a_ω by a gain g_ω. For example, an ideal low-pass filter with cut-off frequency κ is a function, f, that discards signals of frequency $\omega>\kappa$, and passes on signals of frequency $\omega\leq\kappa$, thus:

$$f(e^{i\omega t}):=\begin{cases}e^{i\omega t} & \text{if } \omega\leq\kappa\\ 0 & \text{otherwise.}\end{cases}$$

I.e., the gain g_ω is 0 or 1 according as $\omega\leq\kappa$ or not. To deal with linear filters in our framework, we take the structure $\mathcal{S}$ which parametrises our block diagram language (as described in section 3.2) to provide a name for each subspace of S that is spanned by a finite or co-finite set of the Hilbert basis elements $\mathbf{e}_\omega$. If V and W are such subspaces and if $V\cap W=0$, then, for the purposes of parallel composition, $V\times W$ is identified with the subspace $V+W$ of S.

We now define a simple assertion language appropriate for linear filters (assuming given a language for describing natural numbers, real numbers and sequences and sets thereof). If $\mathbf{x}$ is a signal and $s=(s_1,s_2,\ldots)$ is a sequence of positive real numbers, we write $\mathbf{x}\pm s$ for the box comprising all signals $\mathbf{x}+\sum_{\omega=1}^{\infty}a_\omega\mathbf{e}_\omega$ such that $|a_\omega|\leq s_\omega$ for all ω. If each $s_\omega=\xi$ for some fixed $\xi>0$, we write $\mathbf{x}\pm\xi$ for $\mathbf{x}\pm s$. If b is a box with centre $\mathbf{x}$ then $b=\mathbf{x}+b_0$ for some box b_0 centred at the origin, and if $X\subseteq\mathbb{N}_{>0}$, we write $X\mapsto b$ for the box $\mathbf{x}+\pi_X(b_0)$ where π_X is the orthogonal projection of S onto the subspace S_X spanned by the $\mathbf{e}_\omega$ with $\omega\in X$. Finally, if b and c are boxes and $X,Y\subseteq\mathbb{N}_{>0}$, we write $X\mapsto b\oplus Y\mapsto c$ for the box defined by the following conditions:

$$\begin{aligned}\pi_{X\setminus Y}(X\mapsto b\oplus Y\mapsto c) &= \pi_{X\setminus Y}(X\mapsto b)\\ \pi_{\overline{X\setminus Y}}(X\mapsto b\oplus Y\mapsto c) &= \pi_{\overline{X\setminus Y}}(Y\mapsto c)\end{aligned}$$

where we write $\overline{Z}$ for $\mathbb{N}_{>0}\setminus Z$. The operation $\oplus$ is commutative when $X\cap Y=\emptyset$ and is associative unconditionally.

A real-world linear filter can then be modelled as a parallel composition of idealised primitives $G(X, g, \xi)$, where $X \subseteq \mathbb{N}_{>0}$ is finite or cofinite and $g, \xi \in \mathbb{R}$ with $\xi > 0$, satisfying the following specification for any $0 < \varepsilon_0 \in \mathbb{R}$:

$$\{X \mapsto \mathbf{x} \pm \varepsilon_0\} G(X, g, \xi) \{X \mapsto g\mathbf{x} \pm \xi\varepsilon_0\}$$

Thus $G(X, g, \xi)$ represents a set of ideal filters that kill frequencies outside X, and for each frequency $\omega \in X$ have gain g_ω within a tolerance proportional to ξ.

To reason about linear filters in this framework, note that the sets of filters $G(X, g, \xi)$ can themselves be viewed as parallel compositions: if $X = Y \cup Z$ where $Y \cap Z = \emptyset$, we have:

$$G(X, g, \xi) \quad = \quad \mathsf{Par}(G(Y, g, \xi), G(Z, g, \xi)).$$

So, for example, a real-world low-pass filter might be modelled as the following parallel composition $L(\kappa, \xi, t)$:

$$L(\kappa, \xi, t) \quad := \quad \mathsf{Par}(G(0 \ldots \kappa - t - 1, 1, \xi), G(\kappa - t \ldots \kappa + t, 0.5, 0.5 + \xi), G(\kappa + t + 1 \ldots \infty, 0, \xi))$$

where $\kappa \in \mathbb{N}$, $0 < t \in \mathbb{N}$ and $0 < \xi \in \mathbb{R}$ are parameters. This means that the real-world filter will approximate the ideal low-pass filter within the specified tolerance ξ except for a transitional range of frequencies of width $2t$ centred on the cut-off frequency κ. In the transitional range, the gain is only required to lie approximately in the interval $[0, 1]$. Similarly a real-world high-pass filter with parameters $\lambda \in \mathbb{N}$, $0 < u \in \mathbb{N}$ and $0 < \zeta \in \mathbb{R}$ would be modelled as:

$$H(\lambda, \zeta, t) \quad := \quad \mathsf{Par}(G(0 \ldots \lambda - t - 1, 0, \zeta), G(\lambda - t \ldots \lambda + t, 0.5, 0.5 + \zeta), G(\lambda + t + 1 \ldots \infty, 1, \zeta))$$

With these relations, one may, for example, use the sequence rule to explore the conditions under which the sequential composition: $L(\kappa, t, \xi); H(\lambda, u, \zeta)$ serves as a band-pass filter. Details are left to the reader.

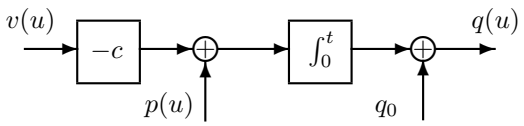
5.3. Steam boiler

The steam boiler problem [ABL96] is a classic example problem in hybrid systems. The steam boiler in this problem involves various physical units such as the pumps that inject water into the boiler and devices to measure the water level in the boiler and the rate of flow of steam out of the boiler. These units are controlled by a digital controller which communicates with them at fixed 5 seconds intervals. The controller has several modes of operation. In the *normal* mode of operation, the physical units are all working and the controller controls the rate of flow of water into the steam boiler by switching the pumps on or off according to the reading obtained from the water level measuring unit. However, the physical units can fail: when that happens the controller enters various failure modes depending on the nature of the failure. Here we consider the *rescue* mode: the controller enters this mode when the water level measuring unit has failed, but other units are working well enough for operation to continue using the other units to estimate the water level. We will give a model for the method of estimation that could be used to assess whether the design is sufficiently accurate for safe operation.

Our model of the steam boiler comprises the steam-boiler itself, a pump (together with an actuator) to provide the steam-boiler with water at a rate $p(t)$ (litres/sec), and a device to measure the rate of steam volume (litres/sec) coming out $v(t)$ of the steam-boiler. We assume that in the *rescue* mode, the controller maintains a record of the last good reading from the water level measuring unit, say q_0, giving our estimate of the water level on the cycle when the unit failed. The volume of water $q(t)$ in the pump at time t can be described via a simple differential equation as

$$q(t) = q_0 + \int_{u=0}^{t} (p(u) - c \cdot v(u)) du$$

where c converts volume of steam into volume of water. This can then be modelled as the following block diagram:



allowing us to estimate the current amount of water $q(t)$ in the steam-boiler. Assume also that the amount of water should never exceed a pre-established maximum value M_1 and never go lower than a minimum value M_2 within five seconds of the water level sensor failing to work. We can try to prove this property in our Hoare logic as follows:

$$\frac{\{\langle p \pm \varepsilon, v \pm \delta\rangle\}(+)\{(p - cv) \pm (\varepsilon + c\delta)\} \quad \Big\{(p - cv) \pm (\varepsilon + c\delta)\Big\}\int_0^t \Big\{(\int_0^t p - cv) \pm t(\varepsilon + c\delta)\Big\}}{\Big\{\langle p \pm \varepsilon, v \pm \delta\rangle\Big\}(1 \times (-c)); (+); \int_0^t ; (+)\Big\{(q_0 + \int p - cv) \pm t(\varepsilon + c\delta)\Big\}} \ (\mathsf{Seq})$$

The derived Hoare triple says that if we can measure the actual values of $p(u)$ and $v(u)$ within error ε and δ, respectively, then we can be sure that the level of water in the steam boiler after time t is $q_0 + \int_0^t p - cv$ within error $t(\varepsilon + c\delta)$. Hence, we can ensure that the water level is within the specified range after 5 seconds given

$$\begin{aligned} q_0 + (\textstyle\int_0^5 p - cv) + 5(\varepsilon + c\delta) &< M_1 \\ q_0 + (\textstyle\int_0^5 p - cv) - 5(\varepsilon + c\delta) &> M_2. \end{aligned}$$

These bounds might be used, for example, as part of a safety case for the steam boiler to determine the conditions under which it is safe to enter the rescue mode.

6. Conclusions and related work

In [MMO06] we have developed an abstract categorical framework for *sound and complete* Hoare logics, called Abstract Hoare Logic, and have shown that several well-known Hoare logics can be viewed as instantiations of it, such as Hoare's original logic for while-programs and the reasoning about pointer-programs by local reasoning. In here we have described a particular sound verification functor for block diagram systems whose denotations are bounded linear operators on Banach spaces. By the general theory developed in [MMO06], our sound verification functor gives rise to a sound Hoare logic for reasoning about Simulink-like block diagrams. There are many directions for further work.

First, we have only just began to explore the richness of theory and tools of linear systems. One next step is to consider questions of completeness and decidability. As we have indicated we have expressed the rules using boxes (i.e. tuples of balls) to represent pre- and post conditions. In practice we envisage these boxes being expressed in some fairly tractable language, where the kind of verification conditions we get can be automatically decided by a theorem prover. We would expect to build on the frameworks provided by ClawZ, built to translate Simulink diagrams into Z and reason about them using the ProofPower proof engine [ACOS00], or on recent work representing block diagrams in Maple, which could be supported by the Maple-PVS engine [GKM05].

Our use of relations suggest strong links between our work and a refinement approach to the design of control systems: a very promising candidate is Cavalcanti's work on Circus [CCO05], which adds CSP and refinement to ClawZ.

Additive relations were originally devised by Mac Lane as a framework for dealing with "diagram chasing" arguments in homology theory. In our context, additive relations provide a simple and convenient alternative to the various notions of partial traced monoidal category that have been proposed, e.g., by Haghverdi and Scott in connection with Geometry of Interaction [HS05]. Other abstract algebraic concepts have also played a broad role in informing research in control engineering, see for example the work of Sontag (Lie Algebras) [Son98] or Willems (relations and syzygies) [PW98]. Rutten [Rut05] proposes stream calculus as a method to give symbolic coinductive representations for signal flow graphs. There is plenty more to be done, both in refining and developing logics directly for such systems, and in using those logics to inform practical developments.

Likewise, our work is part of a rich seam on generalisations of Hoare logic. Amongst the most prominent of these are the algebraic approach of Kozen (dynamic logic and Kleene algebras with test) [Koz97] the tradition of Esik and Bloom [BE93], and the categorical approach of Abramsky et al. (Specification Structures) [AGN95]. Platzer's differential dynamic logic [Pla10] extends dynamic logic with constructs to

describe continuous evolution within a hybrid system using differential equations. Our work focusses on a compositional approach to purely continuous systems in a general framework.

Leitner [Lei08] compares two approaches to verification of models of hardware using model-checking: the Simulink Design Verifier and VIP/SPIN. These approaches concentrate on discrete models and discrete properties and would apply at a lower level of the design process than the continuous models that we consider in the present paper. Sheeran [She05] and others in the functional programming community have developed approaches to functional programming and hardware design based on combinators for serial composition, parallel composition and looping. Again these deal with discrete execution models and discrete properties.

None of the above-mentioned work, however, explicitly exploits linearity. The additional algebraic structure in the approach of the present paper leads both to important simplifications in the metatheory, e.g. simple sound rule for feedback systems, and to many potential benefits to the user, e.g., powerful automated decision procedures.

Acknowledgements. This research was supported by EPSRC grants GR/M98340, GR/L48256, EP/F02309X/1 and by a Royal Society fellowship (516002.K501/RH/kk) to the third author. We are indebted to Hanne Gottliebsen, Henri Huijberts and Graham White for helpful discussions.

References

[ABL96] Jean-Raymond Abrial, Egon Börger, and Hans Langmaack, editors. *Formal Methods for Industrial Applications, Specifying and Programming the Steam Boiler Control*, volume 1165 of *Lecture Notes in Computer Science*. Springer, 1996.

[ACOS00] Rob Arthan, Paul Caseley, Colin O'Halloran, and Alf Smith. ClawZ: Control laws in Z. In *3rd International Conference on Formal Engineering Methods (ICFEM 2000)*, 2000.

[AGN95] Samson Abramsky, Simon J. Gay, and Rajagopal Nagarajan. Specification structures and propositions-as-types for concurrency. In F. Moller and G. Birtwistle, editors, *Logics for Concurrency: Structure vs. Automata—Proceedings of the VIIIth Banff Higher Order Workshop*. Springer-Verlag, 1995.

[AMMO07] Rob Arthan, Ursula Martin, Erik Arne Mathiesen, and Paulo Oliva. Reasoning about linear systems. In *Fifth IEEE International Conference on Software Engineering and Formal Methods SEFM 2007*, pages 123–134. IEEE Press, 2007.

[AMMO09] Rob Arthan, Ursula Martin, Erik Arne Mathiesen, and Paulo Oliva. A general framework for sound and complete Hoare logics. *ACM Transactions onflo Computational Logic*, 11(1):1–31, 2009.

[BE93] Stephen L. Bloom and Zoltán Ésik. *Iteration Theories: the Equational Logic of Iterative Processes*. Springer Verlag, 1993.

[BHM03] Richard J. Boulton, Ruth Hardy, and Ursula Martin. A Hoare logic for single-input single-output continuous-time control systems. In Proceedings 6th International Workshop on Hybrid Systems, Computation and Control, LNCS vol 2623, pages 113–125. Springer, 2003.

[CCO05] Ana L. C. Cavalcanti, Phil Clayton, and Colin O'Halloran. Control law diagrams in *Circus*. In J. Fitzgerald, I. J. Hayes, and A. Tarlecki, editors, *FM 2005: Formal Methods*, volume 3582 of *Lecture Notes in Computer Science*, pages 253 – 268. Springer-Verlag, 2005.

[GKM05] Hanne Gottliebsen, Tom Kelsey, and Ursula Martin. Hidden verification for computational mathematics. *Journal of Symbolic Computation*, 39:539–567, 2005.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10), 1969.

[HS05] Esfandiar Haghverdi and Phil Scott. Towards a typed geometry of interaction. In L. Ong, editor, *CSL'05, LNCS*, volume 3634, pages 216–231. Springer-Verlag, 2005.

[Jon03] Cliff B. Jones. The early search for tractable ways of reasoning about programs. *Annals of the History of Computing*, 25(2), 2003.

[JSV96] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119:447–468, 1996.

[Koz97] Dexter Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997.

[Lei08] Florian Leitner. Evaluation of the Matlab Simulink Design Verifier versus the model checker SPIN. Technical Report soft-08-05, University of Konstanz, 2008.

[Mac75] Saunders Mac Lane. *Homology*, volume 114 of *Der Grundlehren der mathematischen Wissenschaften*. Springer, 1975.

[MMO06] Ursula Martin, Erik A. Mathiesen, and Paulo Oliva. Abstract Hoare logic. *Proceedings of CSL'2006, LNCS 4207*, pages 501–515, 2006.

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010.

[PW98] Jan W. Polderman and Jan C. Willems. *Introduction to Mathematical Systems Theory: A Behavioral Approach*. Springer Verlag, New York, 1998.

[Rut05] Jan J. M. M. Rutten. A tutorial on coinductive stream calculus and signal flow graphs. *Theor. Comput. Sci.*, 343(3):443–481, 2005.

[She05] Mary Sheeran. Hardware design and functional programming: a perfect match. *J. Univ. Comput. Sci.*, 11(7):1135–1158, 2005.

[Son98] Eduardo D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems (Second Edition)*. Springer, New York, 1998.