Reasoning about Linear Systems

Rob Arthan Ursula Martin Erik Arne Mathiesen Paulo Oliva
Department of Computer Science
Queen Mary, University of London
London E1 4NS, UK

{rda, uhmm, erikarne, pbo}@dcs.qmul.ac.uk

Abstract

We consider reasoning about linear systems expressed as block diagrams in a general relational setting. Using the notion of additive relation borrowed from homological algebra, the theory of weakest pre-conditions for these systems turns out to be very tractable and gives simple Hoare-style rules for the block diagram constructors. Many natural choices for the logical language used to express properties of linear systems admit a high degree of automation. We show by example how the rules can be used to inform development of a proof while a decision procedure automates the routine work.

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. 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 aspects of this shortfall, while remaining close to engineering practice, by annotating a block diagram with assertions, and using rules of inference to propagate reasoning about assertions through the diagram. This is in the spirit of Hoare Logic [7], which is the basis of many successful approaches to software verification [9]. The approach should scale well: we can reason about both larger components and primitive blocks in the same framework. As we shall see, the 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, $x = x(t)$ giving the distance of the mass from the wall at time t. It is a graphical representation of the following differential equation:

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

Thinking of the force f as the input to the system, we might ask what input is needed to ensure that the instantaneous acceleration of the cart exceeds c. As we will see in Section 6 the answer may be found by a natural proof process supported by an automatic decision procedure.

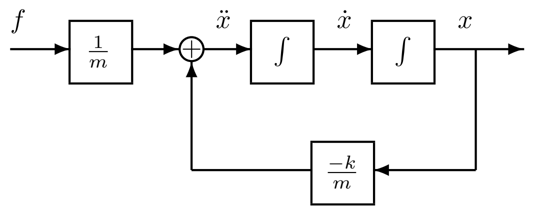

Figure 1. Block Diagram

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 which supports symbolic mathematics, Hoare-style reasoning and Hodes' procedure for quantifier elimination for linear arithmetic over subfields of the reals gives one possibility.

This work was inspired by earlier work [4], 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 [2], 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.

In the next section we give the intuition behind our Hoare logic, and provide some motivating examples. Section 3 contains a more formal review of the basic ideas behind Hoare logic in a relational setting. Section 4 introduces the notion and the semantics for block diagrams. In Section 5 we discuss additive relations, the main technical tool used in the paper. Section 6 describes a Hoare logic for additive relations, and the elements of a soundness proof. Section 7 gives concluding remarks to set the work in its broader context and to sketch future directions for research.

2 Motivating examples

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.

2.1 Matrices and Matrix Composition

Matrices with entries in the field K are a familiar notation for linear transformations. For example, the following matrices r and s act as linear transformations from $\mathbb{R}^3$ to itself: r acts by projection onto the x-y plane while s acts by dilation by a factor of 2 combined with a rotation through $\pi/2$ around the z-axis:

$$r = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad s = \begin{pmatrix} 0 & 2 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

If we consider the unit cube:

$$A :\equiv \{u = (x, y, z) \,|\, 0 \leq x, y, z \leq 1\},$$

then under the action of r, A is mapped to the unit square, B, in the north-west quadrant:

$$B :\equiv \{u = (x, y, 0) \,|\, 0 \leq x, y \leq 1\}.$$

Under the action of s, B is mapped to a larger square, C, in the south-west quadrant:

$$C :\equiv \{u = (x, y, 0) \,|\, 0 \leq x, -y \leq 2\}.$$

So under the action of rs, a point on the surface of the unit cube is mapped to a point in a square of side 2 in the south-west quadrant of the x-y plane.

This is a consequence of the observation that $Ar \subseteq B, Bs \subseteq C$, and hence $Ars \subseteq Bs \subseteq C$. We write $r; s$ for the composition "r followed by s", so in the language of Hoare triples, our deduction can be expressed as the following **sequence rule**:

$$\frac{\{A\}\,r\,\{B\} \quad \{B\}\,s\,\{C\}}{\{A\}\,r; s\,\{C\}}.$$

2.2 Linear Filters

For another example of applications of the sequence rule, consider the combination of two linear filters. A linear filter in the continuous domain is a transformation on signals, which are represented as linear combinations over $\mathbb{R}$ of components of the form $e^{i\omega t}$, where $i^2 = -1$ (often written j in engineering literature), ω is the frequency of the signal, and t is a real variable. Thus signals are elements of the real infinite dimensional vector space spanned by $\{e^{i\omega t} \mid 0 \leq \omega \leq 2\pi\}$. An ideal low-pass filter with cut-off frequency ω_f is a function, F, that discards signals of frequency $\omega > \omega_f$, and passes on signals of frequency $\omega \leq \omega_f$, thus:

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

In our notation this can be expressed as

$$(\omega > \omega_f) \Rightarrow \{e^{i\omega t}\}\,F\,\{0\} \wedge (\omega \leq \omega_f) \Rightarrow \{e^{i\omega t}\}\,F\,\{e^{i\omega t}\}.$$

A sequence of two such filters r, s with cut-off frequencies $\omega_r > \omega_s$ will act as a low pass filter with cut-off frequency ω_s, that is

$$(\omega > \omega_s) \wedge (\omega_r > \omega_s) \Rightarrow \{e^{i\omega t}\}\,Seq(r, s)\,\{0\}.$$

2.3 Differential equations

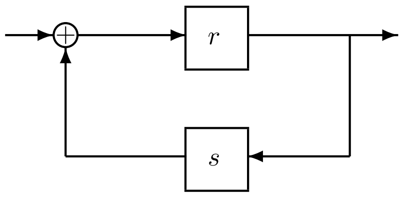

Figure 2. Feedback system

In many applications, it is useful to have feedback loops, as in Figure 2. We write $\mathsf{loop}(r, s)$ for the system obtained from r by feeding its output back to its input through s. Feedback loops represent systems as solutions to equations, e.g., differential equations, such as:

$$x(t) = y(t) + \frac{dy}{dt}.$$

This equation is satisfied by $x(t) = ke^{at}$ and $y(t) = ke^{at}/(1 + a)$, for any $a \neq -1$. Representing the operator d/dt by D, our equation becomes $x/(1 + D) = y$. In order to illustrate the loop rule, let $r = 1, s = -D$ and $A = B = C = V_a = \{ke^{at} \,|\, k \in \mathbb{R}\}$. The general rule has the form

$$\frac{\{B\}\, r\, \{A\} \quad \{A\}\, s\, \{C\}}{\{B - C\}\, \mathsf{loop}(r, s)\, \{A + \mathsf{ker}(r^{-1} - s)\}}$$

subject to some side-conditions described in Section 6.2, which are trivially satisfied in this example. For our example with $r = 1, s = -D$, , we have

- $\{V_a\}\, r\, \{V_a\}$,
- $\{V_a\}\, s\, \{V_a\}$,
- $V_a + \mathsf{ker}(r^{-1} - s) \equiv V_a$,

Applying the loop rule we have, for $a \neq -1$,

$$\{V_a\}\, \mathsf{loop}(1, -D)\, \{V_a\},$$

and thus for any $a \neq -1$, if $x \in V_a$ then $y \in V_a$.

3 Review of Hoare logic

In the sequel we work with systems modelled as (binary) relations between inputs and outputs. We write $r : X \leftrightarrow Y$ to denote that r is a relation between the sets X and Y, i.e., a subset of $X \times Y$, and use $x \underline{r} y$ as a shorthand for $(x, y) \in r$. If r and s are relations, $r; s$ denotes the relational composition r followed by s, so that $x \underline{(r; s)} y$ iff. there is a z with $x \underline{r} z$ and $z \underline{s} y$. If $r : X \leftrightarrow Y$, $r^{-1} : Y \leftrightarrow X$ is the relational inverse of r, defined by taking $x \underline{r^{-1}} y$ iff. $y \underline{r} x$. 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, $\mathsf{ran}(r) = Xr$ is its range and r gives a relation between any sets A and B such that $\mathsf{dom}(r) \subseteq A$ and $\mathsf{ran}(r) \subseteq B$.

Definition 3.1 *If $r : X \leftrightarrow Y$, $A \subseteq X$ and $B \subseteq Y$, a Hoare triple, $\{A\}\, r\, \{B\}$, is a logical judgement which holds whenever $A \subseteq \mathsf{dom}(r)$ and $Ar \subseteq B$.*

The notion of a weakest pre-condition is very useful for working with Hoare triples.

Definition 3.2 *Let $r : X \leftrightarrow Y$ be any relation between sets X and Y and let $B \subseteq Y$. The* **weakest pre-condition of** B **through** r, *$\mathsf{wp}(r, B)$, is defined by any of the following equivalent expressions:*

$$\begin{aligned} \mathsf{wp}(r, B) \quad &:\equiv \quad \{x : \mathsf{dom}(r) \mid \{x\}r \subseteq B\} \\ &= \quad \{x : \mathsf{dom}(r) \mid \forall y : Y \bullet x \,\underline{r}\, y \Rightarrow y \in B\} \\ &= \quad \bigcup \{A : \mathbb{P}X \mid \{A\}\, r\, \{B\}\}. \end{aligned}$$

Thus, the weakest pre-condition of B through r is the set of all points, x, in the domain of r whose image under r is contained entirely in B. Contrast this with the pre-image Br^{-1} of B under r, which comprises all points whose image under r meets B. In general $\mathsf{wp}(r, B)$ is a proper subset of Br^{-1}. But if r is a function (not necessarily total), one has that $\mathsf{wp}(r, B) = Br^{-1}$.

The next proposition shows how Hoare triples are determined by weakest pre-conditions, and how both interact with relational composition.

Proposition 3.3 *i) Let $r : X \leftrightarrow Y$, $A \subseteq X$ and $B \subseteq Y$, then the Hoare triple $\{A\}\, r\, \{B\}$ holds, iff. $A \subseteq \mathsf{wp}(r, B)$.*

ii) Let $r : X \leftrightarrow Y$ and $s : Y \leftrightarrow Z$ be any relations, and let $C \subseteq Z$, then $\mathsf{wp}((r; s), C) = \mathsf{wp}(r, \mathsf{wp}(s, C))$.

iii) Let $r : X \leftrightarrow Y$ and $s : Y \leftrightarrow Z$ be any relations and $A \subseteq X$, $B \subseteq Y$ and $C \subseteq Z$, then the following inference rule for Hoare triples is sound:

$$\frac{\{A\}\, r\, \{B\} \quad \{B\}\, s\, \{C\}}{\{A\}\, r; s\, \{C\}}\,.$$

Proof. *(i)* and *(ii)* are straightforward from the definitions. As for *(iii)*, to say the rule is sound means that the triple below the line must hold whenever the triples above the line hold and this follows easily from *(i)* and *(ii)*. □

Note that B in $\{A\}\, r\, \{B\}$ is a set, not a relation. In general, one expects the predicate represented by B to be parametrised, so that the Hoare triple can be used to define properties such as "the output is twice the input". In the present context, it is adequate to work with sets, since there is a cartesian product construction available which can be used to carry additional information about inputs to the outputs (cf. the use of auxiliary variables in program specification).

4 Signal flow graphs and block diagrams

In this section we define two 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.

4.1 Signal flow graphs

We view signal flow graphs as a notation for describing relations between vector spaces. Most of what follows will work for vector spaces over more general fields but for simplicity we take K to be the field of real numbers, $\mathbb{R}$ in the sequel. So, from now on all vector spaces will be real vector spaces.

As in the examples above, a signal flow graph is a directed graph with two distinguished nodes, the input node and the output node. Each node is labelled with a vector space and each edge with a linear transformation between the vector spaces that label its start and end nodes. A flow will assign to each node of a value in the corresponding vector space satisfying compatibility conditions derived from the linear transformations. The set of all flows will define an input-output relation between the vector spaces labelling the input and output node.

It is technically convenient to set up signal flow graphs so there can be more than one edge between a pair of nodes. We thus define a **signal flow graph** G to be a structure

$$G \equiv (N, E, \phi, \{V_n \mid n \in N\}, \{f_e \mid e \in E\}, I, O)$$

where:

1. N and E form the nodes and edges of a non-empty directed graph, where the **adjacency function** $\phi : E \to N \times N$ maps an edge to the pair (s_e, t_e) comprising its source node, s_e, and target node, t_e;
2. For each node $n \in N$, V_n is a real vector space;
3. For each edge $e \in E$, f_e is a linear transformation from V_{s_e} to V_{t_e}, where s_e and t_e are the source and target nodes of the edge e (i.e., where $\phi(e) = (s_e, t_e)$);
4. I and O, which are referred to as the **input** and **output** nodes respectively, are two distinct nodes, (i.e., $I \in N$, $O \in N$ and $I \neq O$).

Note that the input node I is not required to be a source, the output node O is not required to be a sink and I and O might lie in separate connected components of the underlying undirected graph.

4.2 Flows on signal flow graphs

We now describe how a signal flow graph defines a relation between the vector spaces that label its input and output nodes. But first, a note on notation: from now on, functions, particularly functions on vector spaces, will often be written on the right of their arguments: i.e., if f is a function, the value of f at x may be written as $f(x)$ or as xf (as one would if x were a row vector with m elements and f were an $m \times n$ matrix acting on it).

Now, let $G = (N, E, \phi, V_n, f_e, I, O)$ be a signal flow graph. An assignment to each node $n \in N$, of an element, γ_n, of the vector space V_n is said to be a **flow** with **input** $x \in V_I$ iff. for every node, n, $\gamma_n = x_n + \sum_{\phi(e)=(m,n)} \gamma_m f_e$, where $x_n = 0$ unless $n = I$ and $x_I = x$. I.e., the linear transformation f_e that labels an edge e of the graph is applied to the value at the source node of e and the result contributes to the value at the targetarget node of e; the input value x also contributes to the input node I; the incoming contributions at each node n must then sum to the value γ_n labelling n.

We define the **output** of a flow to be the value γ_O that it assigns to the output node and then define the **input-output relation** $r_G : V_I \leftrightarrow V_O$ of the signal flow graph G by $x \ \underline{r_G}\ y$ iff. there is a flow on G with input x and output y. Note that when the input $x = 0$, one may form a flow with $\gamma_n = 0$ for each n, so the input-output relation of a signal flow graph is non-empty.

For example, the input-output relations of the four signal flow graphs shown in Figure 3 are as follows:

i) $\{(x, y) \mid y = \alpha x\}$,

ii) $\{(x, y) \mid y = \alpha\beta x\}$,

iii) $\{(x, y) \mid (1 - \alpha\beta)y = \alpha x\}$,

iv) $\{(x, y) \mid (1 - \alpha\beta)y = \alpha x\}$.

A flow can be viewed as an element of the cartesian product vector space $\prod_{n \in N} V_n$ 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*, a notion that we will study in more detail in Section 5 below.

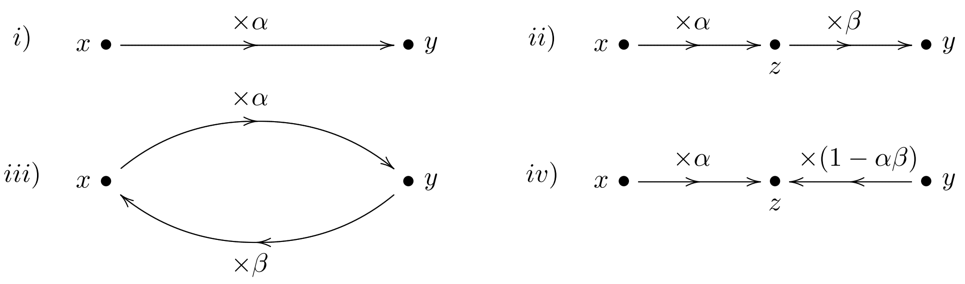

Figure 3. Some Small Signal Flow Graphs

4.3 Structured signal flow graphs

We now specify a subclass of signal flow graphs that are built up recursively from simple blocks using standard constructions. In the following, whenever two signal flow graphs are combined, we assume that edges and nodes are relabelled as necessary to avoid any accidental identifications. We may then identify selected pairs of nodes, one from each of the two graphs, provided the labels on identified nodes agree.

Definition 4.1 *Block diagrams are formed as follows:*

(i) Given a linear transformation $f : V \to W$ we have the **unit block***, $Unit(f)$: it has two nodes I and O labelled with V and W respectively and a unique edge from I to O labelled with f.*

(ii) Let B and C be block diagrams with input nodes I_B, I_C and output nodes O_B, O_C and such that $V_{O_B} = V_{I_C}$ The **sequential composition***, $Seq(B, C)$ of B and C is formed by identifying the output node of B with the input node of C.*

(iii) Let B and C be block diagrams with input nodes I_B, I_C and output nodes O_B, O_C and such that $V_{I_B} = V_{I_C}$ and $V_{O_B} = V_{O_C}$. The **sum***, $Sum(B, C)$ of B and C is formed by identifying I_B with I_C to form the input node and O_B with O_C to form the output node.*

(iv) Let B and C be block diagrams with input nodes I_B, I_C and output nodes O_B, O_C and such that $V_{I_B} = V_{O_C}$ and $V_{O_B} = V_{I_C}$. The **loop***, $Loop(B, C)$ of B and C is formed by identifying I_B with O_C to form the input node and identifying O_B with I_C to form the output node.*

We define BD to be the class of block diagrams built up recursively using the constructs $Unit$, Seq, Sum and $Loop$. Thus BD generalises the Cosy language defined in [4] by allowing arbitrary vector spaces. In implementing tools, one might well want to restrict the class of linear transformations f used in constructing the atomic block diagrams $Unit(f)$, e.g., to linear transformations $\mathbb{R}^p \to \mathbb{R}^q$ given explicitly by matrices.

There are many signal flow graphs which are not block diagrams in this sense, but, as we shall see in the next section 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.

5 Additive relations

Additive relations were designed for reasoning about the "diagram-chasing" methods of homological algebra, [11]. Signal flow graphs are sufficiently close in spirit to the commutative diagrams of homological algebra for additive relations to give a very nice conceptual framework for their semantics.

Definition 5.1 *Let V and W be vector spaces. An* **additive relation** *between V and W is any subspace of the direct sum $V \oplus W$. Equivalently, an additive relation is any non-empty relation, $r : V \leftrightarrow W$, such that if $v_1, v_2 \in V$, $w_1, w_2 \in W$, and $v_1 \ \underline{r}\ w_1$ and $v_2 \ \underline{r}\ w_2$, then also $\alpha v_1 + \beta v_2 \ \underline{r}\ \alpha w_1 + \beta w_2$ for any $\alpha, \beta \in \mathbb{R}$.*

For example, (the graph of) any linear transformation $f : V \to W$ is an additive relation between V and W.

Definition 5.2 *If $r : V \leftrightarrow W$ is an additive relation, the* **kernel** *and* **indeterminacy** *of r are defined by $\mathsf{ker}(r) = \{v : V \mid v \ \underline{r}\ 0\}$ and $\mathsf{ind}(r) = \{w : W \mid 0 \ \underline{r}\ w\}$, respectively.*

Lemma 5.3 *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 $\{z \mid \exists x : X, y : Y \bullet z = x + y\}$. If $x \in V$, we write $x + X$ for $\{x\} + X$. If X happens to be a subspace of V, then sets of the form $x+X$ are referred to as **cosets** of the subspace X and the collection of all cosets 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 5.4 *Let $r : V \leftrightarrow W$ be an additive relation and assume $v \ \underline{r}\ w$, then*

$$\begin{aligned}\{v_1 : V \mid v_1 \ \underline{r}\ w\} &= v + \mathsf{ker}(r),\\ \{w_1 : W \mid v \ \underline{r}\ w_1\} &= w + \mathsf{ind}(r).\end{aligned}$$

Proof. Assume $v_1 \ \underline{r}\ w$, then by the additivity of r, $v_1 - v \ \underline{r}\ w - w = 0$, so $v_1 - v \in \mathsf{ker}(r)$, i.e., $v_1 \in v + \mathsf{ker}(r)$, proving the left-to-right direction of the first equation. The rest is similar. □

The next proposition gives a kind of normal form for additive relations.

Proposition 5.5 *Let $r : V \leftrightarrow W$ be an additive relation, then there is a linear transformation $g : \mathsf{ran}(r) \rightarrow U$ such that $r = (f; g^{-1}) : \mathsf{dom}(r) \leftrightarrow \mathsf{ran}(r)$, where $U = \mathsf{dom}(r)/\mathsf{ker}(r)$ and $f : \mathsf{dom}(r) \rightarrow U$ is the natural projection onto the quotient space U.*

Proof. Recall that the natural projection $f : \mathsf{dom}(r) \rightarrow U$ is defined by $f(v) = v + \mathsf{ker}(r)$. If $w \in \mathsf{ran}(r)$, then, by definition, there is a v_1 such that $v_1 \ \underline{r}\ w$, and, by Lemma 5.4, $\{v \mid v \ \underline{r}\ w\} = v_1 + \mathsf{ker}(r)$. It follows that we may define $g : \mathsf{ran}(r) \rightarrow U$ by $g(w) = \{v \mid v \ \underline{r}\ w\}$. From these definitions, it is easy to check, using Lemma 5.4, 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 $p : V \rightarrow C$ and $q : W \rightarrow D$ say, If one takes $f^* = f \times p \times 0 : V \times V \times V \rightarrow U \times C \times D$ and $g^* = g \times 0 \times q : W \times W \times W \rightarrow U \times C \times D$, one has for any $v \in V$ and any $w \in W$, $v \ \underline{r}\ w$ iff. $vf^* = wg^*$. 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. We now define two other methods for constructing new relations from old ones, which preserve additivity and correspond to the block diagram constructions.

Definition 5.6 *Let V and W be real vector spaces.*

(i) If $r, s : V \leftrightarrow W$, then the **sum** *of r and s, written $r+s$, is defined by taking $v \ \underline{r+s}\ w$ iff. there exist w_1, w_2 such that $v \ \underline{r}\ w_1$, $v \ \underline{s}\ w_2$ and $w = w_1 + w_2$.*

(ii) If $r : V \leftrightarrow W$ and $t : W \leftrightarrow V$, then the **loop** *of r and t, written $\mathsf{loop}(r, t)$, is defined by taking $v \ \underline{\mathsf{loop}(r, t)}\ w$, iff. there exists v_1 such that $w \ \underline{t}\ v_1$ and $(v + v_1) \ \underline{r}\ w$.*

Define $\mathcal{B}$ as the smallest class of relations containing all the linear transformations of real vector spaces and closed under relational composition, sum and loop.

Note that the definition of sum generalises the usual pointwise sum of linear transformations, and is not the direct sum of subspaces of $V \times W$.

We may also define various derived constructions, e.g., scalar multiplication: if $\alpha \in \mathbb{R}$ and $r : V \leftrightarrow W$ is an additive relation, we write αr for the composite: $((v : V \mapsto \alpha v); r)$. In particular, taking $\alpha = -1$, we have negation: $x \ \underline{-r}\ y$ iff. $x \ \underline{r}\ {-y}$.

It is easy to see that if $r, s : V \leftrightarrow W$ and $t : V \leftrightarrow W$ are additive relations, then so are $-r$, $r + s$ and $\mathsf{loop}(r, t)$. Thus $\mathcal{B}$ is a set of additive relations. The following proposition will enable us to prove that $\mathcal{B}$ contains all the additive relations.

Proposition 5.7 *If $r : V \leftrightarrow W$ and $t : V \leftrightarrow W$ are relations and $g : V \rightarrow V$ is a linear transformation, then*

$$\begin{aligned}\mathsf{loop}(r, t) &= (r^{-1} - t)^{-1},\\ g^{-1} &= \mathsf{loop}(1_V, 1_V - g),\\ 1_{\mathsf{ran}(g)} : V \leftrightarrow V &= (\mathsf{loop}(1_{V_1}, 1_V - g); g).\end{aligned}$$

Proof. For the first equation, one has that $v \ \underline{\mathsf{loop}(r, t)}\ w$, iff. $v + v_1 \ \underline{r}\ w$ and $w \ \underline{t}\ v_1$ for some v_1, iff., $w \ \underline{r^{-1}}\ v + v_1$ and $w \ \underline{t}\ v_1$, iff. $w \ \underline{r^{-1} - t}\ v$, iff. $v \ \underline{(r^{-1} - t)^{-1}}\ w$. Since $g^{-1} = (1_V^{-1} - (1_V - g))^{-1}$, the second equation follows from the first taking $W = V$, $r = 1_V = 1_V^{-1}$ and $t = g$. Finally the third equation follows from the second and the fact that for any function g, $(g^{-1}; g) = 1_{\mathsf{ran}(g)}$. □

Theorem 5.8 *$\mathcal{B}$ comprises all additive relations.*

Proof. Let $r : V \leftrightarrow W$ be any additive relation, and let U, $f : \mathsf{dom}(r) \rightarrow U$ and $g : \mathsf{ran}(r) \rightarrow U$ be as in Proposition 5.5, so that $r = (f; g^{-1}) : \mathsf{dom}(r) \leftrightarrow \mathsf{ran}(r)$. Up to an isomorphism, one may assume that $U \subseteq W$; then, choosing bases for V and W that extend bases for $\mathsf{dom}(r)$ and $\mathsf{ran}(r)$ respectively, one can extend f and g to linear transformations $\hat{f} : V \rightarrow W$ and $\hat{g} : W \rightarrow W$ such that r factors as $r = (1_{\mathsf{dom}(r)}; \hat{f}; \hat{g}^{-1}; 1_{\mathsf{ran}(r)})$:

$$V \xrightarrow{1_{\mathsf{dom}(r)}} V \xrightarrow{\hat{f}} W \xrightarrow{\hat{g}^{-1}} W \xrightarrow{1_{\mathsf{ran}(r)}} W.$$

Using this factorisation, it follows from Proposition 5.7 and the fact that any subspace of a vector space V can be written as $\mathsf{ran}(h)$ for some linear transformation $h : V \to V$ that $r \in \mathcal{B}$. $\square$

Now, if B and C are members of the class BD of block diagrams defined in Section 4.3 above, with input-output relations r_B and r_C, then the input-output relations of the block diagrams $Seq(B, C)$, $Sum(B, C)$ and $Loop(B, C)$ are $(r_B; r_C)$, $r_B + r_C$ and $\mathsf{loop}(r_B, r_C)$ respectively. Thus $\mathcal{B}$ is precisely the class of all relations that can be realised as the input-output relation of a block diagram. Note also that, by Proposition 5.7, that loop and relational inverse are interdefinable, so we may take relational inverse as an alternative primitive constructor for block diagrams, if we so please, and this will often simplify our algebra.

It is not hard to see using Lemma 5.4 that the additive relation r induces a vector space isomorphism between the subquotient $\mathsf{dom}(r)/\mathsf{ker}(r)$ of V and the subquotient $\mathsf{ran}(r)/\mathsf{ind}(r)$ of W, and, conversely, any isomorphism between a subquotient V_1/V_2 of V and a subquotient W_1/W_2 of W induces an additive relation between V and W. Thus additive relations are quite like functions and so weakest pre-condition calculation should work out nicely for them, as the following lemma shows by giving a simple general description of the weakest pre-condition through an additive relation. To form the weakest pre-condition $\mathsf{wp}(r, B)$, first discard from B any element w such that $w + \mathsf{ind}(r)$ is not contained in B, $\mathsf{wp}(r, B)$ is the the pre-image through r of what remains.

Proposition 5.9 *Let $r : V \leftrightarrow W$ be an additive relation and let $B \subseteq W$. Let $B_0 = \{b : B \mid b + \mathsf{ind}(r) \subseteq B\}$, then:*

i) $\mathsf{wp}(r, B) = B_0 r^{-1}$,

ii) If $B = Ar$ for some $A \subseteq V$, then $\mathsf{wp}(r, B) = A + \mathsf{ker}(r)$.

Proof. *i)* If $v_0 \in \mathsf{wp}(r, B)$, then certainly $v_0 \in \mathsf{dom}(r)$ and there is a $b \in B$ such that $v_0 \ \underline{r}\ b$. Now if $w \in \mathsf{ind}(r)$, i.e., $0 \ \underline{r}\ w$, then as r is additive, one has that $v_0 = v_0 + 0 \ \underline{r}\ b + w$, but this implies that $b + w \in B$, since $v_0 \in \mathsf{wp}(r, B)$. Thus $b \in B_0$. If $v_0 \in B_0 r^{-1}$, there is a b such that $b + \mathsf{ind}(r) \subseteq B$ and $v_0 \ \underline{r}\ b$. But then by Lemma 5.4, if $v_0 \ \underline{r}\ w$, $w \in B + \mathsf{ind}(r) \subseteq B$ which completes the proof of *(i)*. *ii)* if $B = Ar$, and $b \in B$, then there is an $a \in A$ such that $a \ \underline{r}\ b$, but then $a \ \underline{r}\ b + x$ for any $x \in \mathsf{ind}(R)$, so in part *(i)* we have $B_0 = B$ and $\mathsf{wp}(r, B) = Br^{-1} = Arr^{-1} = A + \mathsf{ker}(r)$. $\square$

In applications of the above proposition it is often useful to note that if $X = Yr^{-1}$, then $X + \mathsf{ker}(r) = X$ and if $Y = Xr$, then $Y + \mathsf{ind}(r) = Y$ (i.e., passing a set one way or another through an additive relation produces a set that is homogeneous with respect to the kernel or indeterminacy).

6 A Hoare logic for additive relations

We propose to use Hoare triples $\{A\}\, r\, \{B\}$ as a specification notation for additive relations r, with assertions A and B about the input/output relation. In a practical implementation, the additive relations might be given using matrices and the block diagram constructors, and the pre- and post-conditions A and B would be given using some logical language. In this section we discuss one such language: the language of linear real arithmetic.

6.1 A logical language for linear algebra

The language L of linear real arithmetic is the first order language of the ordered ring $\mathbb{R}$ (with operators $\_+\_$, $-\_$ and $\_\times\_$ and predicate symbols $\_=\_$ and $\_<\_$) with the multiplication operator $\_\times\_$ restricted so that the left operand must be a constant, L being taken to include a constant α for each real number α. If K is a subfield of $\mathbb{R}$, we also have the language L_K of linear arithmetic over K, in which the constants c are only provided for $c \in K$. Note that after multiplying out and collecting like terms an expression of L is just a polynomial, $\alpha_1 x_1 + \ldots + \alpha_n x_n + \gamma$, of degree 1 in the variables $x_1, \ldots, x_n$. An atomic formula of L may then be rearranged into a normal form $\alpha_1 x_1 + \ldots + \alpha_n x_n = \gamma$ or $\alpha_1 x_1 + \ldots + \alpha_n x_n < \gamma$, where γ and the α_i are constants.

$\mathcal{PL}_n$ is then the set of all subsets of $\mathbb{R}^n$ of the form: $\{(x_1, \ldots, x_n) \mid \mathcal{F}\}$ where $\mathcal{F}$ is a formula of L whose free variables are contained in $\{x_1, \ldots, x_n\}$. The set $\mathcal{PL}$ of **piecewise linear sets** is the union of the $\mathcal{PL}_n$, $n \in \mathbb{N}$.

Proposition 6.1 *If $B \in \mathcal{PL}_n$ and $r : \mathbb{R}^m \leftrightarrow \mathbb{R}^n$ is an additive relation, then* $\mathsf{wp}(r, B) \in \mathcal{PL}_m$.

Proof. By the remarks following Proposition 5.5, 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. $\mathbf{vF} = \mathbf{wG}$. We then have

$$\begin{array}{rclll} \mathsf{wp}(r, B) & = & \{\mathbf{v} & \mid & (\exists \mathbf{w} \bullet \mathbf{vF} = \mathbf{wG}) \\ & & & \wedge & (\forall \mathbf{w} \bullet \mathbf{vF} = \mathbf{wG} \Rightarrow \mathbf{w} \in B\}. \end{array}$$

The matrix equations here are equivalent to a formula in L and, as B is in $\mathcal{PL}_n$, the formula $\mathbf{w} \in B$ is also equivalent to a formula in L, so $\mathsf{wp}(r, B) \in \mathcal{PL}_m$. $\square$

The following classic result of Hodes [8] about quantifier elimination for the languages L_K allows us to go further:

Proposition 6.2 *Every formula of L_K is equivalent to a quantifier-free formula of L_K.*

Proof. Sketch (for details see [8]). The proof gives a procedure which given a formula $\mathcal{F}$ constructs a quantifier-free equivalent, and this procedure is effective (i.e., an algorithm) if computations can be effectively carried out in the subfield K of $\mathbb{R}$, e.g., if K is the field of rational numbers. By standard logical arguments and using the fact that $\neg x = y$ in $\mathbb{R}$ iff. $x < y \vee y < x$, one can reduce to the case where $\mathcal{F}$ has the form $\exists x_1 \bullet \mathcal{E}$ where $\mathcal{E}$ is a conjunction of atomic formulae, each of which we may assume to be in the normal form mentioned above and each of which contains a (non-zero) free occurrence of x_1:

$$\Sigma_{i=1}^{n} \alpha_{1i} x_i \ \underline{r_1} \ \gamma_1 \wedge \ldots \wedge \Sigma_{i=1}^{n} \alpha_{ki} x_i \ \underline{r_k} \ \gamma_k,$$

where $r_i \in \{=, <\}$. If there is an equation, we may scale and rearrange it into the form $x_1 = e$, where x_1 does not appear in e and then eliminate the quantifier by discarding the equation and replacing x_1 by e in the remaining atomic formulae. If there are only inequalities, then we may scale and rearrange the j-th inequality to have the form $e_j < x_1$ or $x_1 < e_j$. Let I be the set of those j where the j-th inequality gives a lower bound $e_j < x_1$ on x_1 and J be the set of those j where the j-th inequality gives an upper bound $x_1 < e_j$, then there exists an x_1 satisfying all these bounds iff. $e_i < e_j$ for every $i \in I$ and $j \in J$, so we may eliminate the quantifier in favour of the conjunction of these inequalities. $\square$

Hodes' theorem has nice consequences. Geometrically it implies that any piecewise linear set is a union of generalised polyhedra. Logically, when K is a subfield of $\mathbb{R}$ admitting effective arithmetic computation, it means that truth in the language L_K is decidable: to decide the truth of a formula, form its universal closure and then eliminate quantifiers resulting in an equivalent formula containing no variables whose truth can be decided by arithmetic computation.

By standard transformations of set-theory into predicate calculus, Hodes' procedure provides a decision procedure and a normal form for an extended language including the standard set theory notations and notations such as $X + Y$, where X and Y are subsets of $\mathbb{R}^n$ that can be defined using set comprehension and linear arithmetic.

6.2 The Hoare logic

Our Hoare logic is developed relative to a decision procedure used for checking simple Hoare triple. Therefore, as axioms of our Hoare logic we take some set of Hoare triples $\{A\}\, r\, \{B\}$ whose validity can assessed by the decision procedure.

We now develop inference rules for reasoning about more complex Hoare triples, $\{A\}\, r\, \{B\}$, where $r : V \leftrightarrow W$ is an additive relation on the real vector spaces V and W and A and B are subsets of V and W respectively. The rules are sound, but we do not claim that they are complete. We do however, show by example that they allow non-trivial properties of systems to be derived.

We have expressed the rules using arbitrary sets to represent the pre- and post-conditions, but in practice we envisage the pre- and post-conditions being expressed in some fairly tractable first-order language (such as the language of linear arithmetic over a decidable subfield of the reals). In such a context, one would expect to be able to formulate and prove an appropriate completeness result.

As always, we have the sequence rule of Proposition 3.3. There are also simple rules for linear combinations of assertions and relations:

Proposition 6.3 (Soundness Theorem) *Let $r, s : V \leftrightarrow W$ and $t : W \leftrightarrow U$ be additive relations; let $A \subseteq V$, $B, B_1 \subseteq W$, $C \subseteq U$ and $\beta, \gamma \in \mathbb{R}$ then the* **linear combination rule**

$$\frac{\{A\}\, r\, \{B\} \quad \{A\}\, s\, \{B_1\}}{\{A\}\, \beta r + \gamma s\, \{\beta B + \gamma B_1\}} \ ,$$

the **inverse rule**

$$\frac{\{A\}\, r\, \{B\}}{\{B\}\, r^{-1}\, \{A + \mathsf{ker}(r)\}} \ B \subseteq Ar,$$

and the **sequence rule**

$$\frac{\{A\}\, s\, \{B\} \quad \{B\}\, t\, \{C\}}{\{A\}\, s; t\, \{C\}}$$

are sound (subject to the stated side-condition in the case of inverse rule).

Proof. The soundness of the rules for negation and linear combination is easy to check from the definitions. For the rule for inverse, assume that $\{A\}\, r\, \{B\}$ holds, i.e., that $A \subseteq \mathsf{dom}(r)$ and $Ar \subseteq B$, and assume also that the side-condition $B \subseteq Ar$ holds, so that in fact $Ar = B$, so that $Br^{-1} = A(r; r^{-1}) = A + \mathsf{ker}(r)$. Applying Proposition 5.9, we have that $\mathsf{wp}(r^{-1}, A + \mathsf{ker}(r)) = Br^{-1}r = B + \mathsf{ind}(r)$, but as $Ar = B$, $B + \mathsf{ind}(r) = B$ and we are done by Proposition 5.9. $\square$

As mentioned above, the construction of feedback can be derived from the construction of inverse relations. That also gives us a way of obtaining a Hoare logic rule for the feedback from that of the inverse given above. The rule for the feedback is the **loop rule**:

$$\frac{\{B\}\, r\, \{A\} \quad \{A\}\, s\, \{C\}}{\{B - C\}\, \mathsf{loop}(r, s)\, \{A + \mathsf{ker}(r^{-1} - s)\}}$$

subject to the side conditions:

(a) $A \subseteq Br$

(b) $B + \mathsf{ker}(r) \subseteq B$

(c) $B - C \subseteq A(r^{-1} - s)$

$$\begin{tikzcd}[column sep=huge]
\bullet \arrow[r, "{\begin{pmatrix} \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{m} \end{pmatrix}}"] & \bullet \arrow[r, "{\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}}"] & \bullet \arrow[r, "{\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}}"] & \bullet \arrow[ll, bend left=60, "{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\frac{k}{m} & 0 & 0 \end{pmatrix}}"]
\end{tikzcd}$$

Figure 4. A Simple Control System

This rule is derivable as:

$$\dfrac{\dfrac{\{A\}\,s\,\{C\} \quad \dfrac{\{B\}\,r\,\{A\}}{\{A\}\,r^{-1}\,\{B\}}}{\{A\}\,r^{-1} - s\,\{B - C\}}}{\{B - C\}\,(r^{-1} - s)^{-1}\,\{A\}}$$

given the side-conditions (a), (b) and (c).

Let us consider as an example the system of Figure 1. We will use the proof rules for guidance and to suggest conjectures and then use a decision procedure to check side-conditions. A block diagram for the system is shown in Figure 4: we have translated from a differential equation to a matrix representation via a state space representation. The figure shows a block diagram with $V_n = \mathbb{R}^3$ for each node n. The input-output relation of the block diagram is given by the additive relation $s = (f; \mathsf{loop}((g; g), h))$, where $f, g, h : \mathbb{R}^3 \to \mathbb{R}^3$ are given by:

$$f = \frac{1}{m} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad g = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$h = -\frac{k}{m} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

where m and k are positive constants. For given c, let $A = \{(x, y, z) \mid z > c\}$, we will derive the weakest precondition $\mathsf{wp}(s, A)$ using the proof rules to guide us.

Putting $r = (g; g)^{-1}$ and applying Proposition 5.7, we have that $s = (f; (r - h)^{-1})$. Take $X = Ar = \{(x, y, z) \mid x > c\}$ and $Y = Ah = \{(x, 0, 0) \mid -\frac{m}{k}x > c\}$ and consider the following proof tree:

$$\dfrac{\dfrac{\{A\}\,r\,\{X\} \quad \{A\}\,h\,\{Y\}}{\{A\}\,r - h\,\{X - Y\}}}{\{X - Y\}\,(r - h)^{-1}\,\{A + \mathsf{ker}(r - h)\}}\ .$$

The side-condition for the final inference requires that $X - Y \subseteq A(r - h)$, as may be verified using Hodes' procedure. Applying Hodes' procedure to $X - Y$, one finds that $X - Y = \{(x, y, z) \mid x > (1 + \frac{k}{m})c\}$, and then one gets the pre-condition $(X - Y)f^{-1} = \{(x, y, z) \mid x > (m + k)c\}$ for A through s. That this is in fact the weakest pre-condition may be verified by yet another application of the Hodes' procedure.

7 Conclusions and related work

We have shown that we can use additive relations to construct a Hoare Logic for block diagrams over an arbitrary vector space, easily expressed using standard linear algebra. 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 arbitrary sets, over possibly infinite dimensional spaces, to represent pre- and post conditions. In practice we envisage these sets being expressed in some fairly tractable first order language, such as the language of linear arithmetic over a decidable subfield of the reals. In this context it should be fairly straightforward to devise a completeness result. If we restrict to the finite-dimensional case we might restrict the pre and post conditions, for example, to be piecewise linear, where we can use standard algorithms of linear algebra to get decidability. A more expressive approach would be to use semi-algebraic sets, which correspond to using the full first order language of the reals to express assertions. This is still decidable, though the decision procedures are provably much less efficient.

Once resolved these questions will guide the most appropriate practical implementation, which we would expect to draw upon the techniques of both computational linear algebra and decision procedures over the reals. 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 [2], or on recent work representing block diagrams in Maple, which could be supported by the Maple-PVS engine [6].

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 [5], 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. 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) [15] or Willems (relations and syzygies) [13]. Rutten [14] 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 (Kleene Algebras with Test) [10] in the tradition of Esik and Bloom [3], and the categorical approach of Abramsky et al. (Specification Structures) [1].

In [12] 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 our setting, since $\mathcal{B}$ comprises all the additive relations, we can add any additional operators for constructing new additive relations from old ones. In particular, given any additive relations $r : V_1 \leftrightarrow W_1$ and $s : V_2 \leftrightarrow W_2$, one can form the cartesian product relation $r \times s : V_1 \times V_2 \leftrightarrow W_1 \times W_2$ defined so that $(v_1, v_2)\ \underline{r \times s}\ (w_1, w_2)$ iff. $v_1\ \underline{r}\ w_1$ and $v_2\ \underline{s}\ w_2$. If $V_1 = V_2 = V$, say, one also has the pairing (r, s) defined such that $v\ \underline{(r, s)}\ (w_1, w_2)$ iff. $v\ \underline{r}\ w_1$ and $v\ \underline{s}\ w_2$. With the monoidal structure arising from cartesian product $\mathcal{B}$ becomes a traced monoidal category with the trace defined by the loop operator. Thus the Hoare logic developed here can be viewed as an instantiation of the framework developed in [12], if one relaxes the condition on the verification functor aiming only at soundness.

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., by admitting a simple general description of weakest pre-conditions, and to many potential benefits to the user, e.g., powerful automated decision procedures.

Acknowledgements. This research was supported by EPSRC grants GR/M98340 and GR/L48256, and by an EPSRC studentship to the third author. We are indebted to Hanne Gottliebsen, Henri Huijberts and Graham White for helpful discussions and to the anonymous referees for their comments.

References

[1] S. Abramsky, S. J. Gay, and R. 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.

[2] R. D. Arthan, P. Caseley, C. O'Halloran, and A. Smith. ClawZ: Control laws in Z. In *3rd International Conference on Formal Engineering Methods (ICFEM 2000)*, 2000.

[3] S. L. Bloom and Z. Esik. *Iteration theories: the equational logic of iterative processes*. Springer Verlag, 1993.

[4] R. J. Boulton, R. Hardy, and U. 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.

[5] A. L. C. Cavalcanti, P. Clayton, and C. 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.

[6] H. Gottliebsen, T. Kelsey, and U. Martin. Hidden verification for computational mathematics. *Journal of Symbolic Computation*, 39:539–567, 2005.

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

[8] L. Hodes. Solving problems by formula manipulation in logic and linear inequalities. *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, pages 553–559, 1971.

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

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

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

[12] U. Martin, E. A. Mathiesen, and P. Oliva. Abstract Hoare logic. *Proceedings of CSL'2006, LNCS 4207*, pages 501–515, 2006.

[13] J. Polderman and J. Willems. *Introduction to Mathematical Systems Theory: A Behavioral Approach*. Springer Verlag, New York, 1998.

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

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