An Abstract Block Formalism for Engineering Systems*

Ievgen Ivanov[1,2]

[1]Taras Shevchenko National University of Kyiv, Ukraine
[2]Paul Sabatier University, Toulouse, France

ivanov.eugen@gmail.com

Abstract. We propose an abstract block diagram formalism based on the notions of a signal as a time-varying quantity, a block as a signal transformer, a connection between blocks as a signal equality constraint, and a block diagram as a collection of interconnected blocks. It does not enforce implementation details (like internal state-space) or particular kinds of dynamic behavior (like alternation of discrete steps and continuous evolutions) on blocks and can be considered as an abstraction of block diagram languages used by engineering system designers.
We study its properties and give general conditions for well-definedness of the operation of a system specified by a block diagram for each admissible input signal(s).

Keywords. block diagram, signal transformer, semantics, engineering system

Key Terms. Mathematical Model, Specification Process, Verification Process

1 Introduction

Many software tools for developing control, signal processing, and communication systems are based on block diagram notations familiar to control engineers. Examples include system design software Simulink [1], Scicos [2], Dymola [3], SCADE [4], declarative synchronous languages [5], some embedded programming languages [6, 7].

In such notations, a diagram consists of blocks (components) connected by links. Typically, blocks have input and output ports, and (directed) links connect output ports of one block with input ports of the same or another block. A block is interpreted as an operation which transforms input signals (i.e. time-varying quantities) flowing though its input ports into output signals.

* Part of this research has been supported by the project *Verisync* (ANR-10-BLAN-0310), France.

Wide applicability of block diagram notations makes them an interesting object of study from a theoretical perspective. Classical control theory and signal processing already provide some degree of formal treatment of block diagrams [8, 9], but this is normally not sufficient to handle such aspects of modern system design languages as mixing of analog and discrete-time blocks, partially defined block operations, non-numeric data processing, etc.

To take these issues into account, researches developed formal semantics for various block diagram languages [10–13]. Some effort has been made to unify approaches taken by different engineering system modeling and analysis tools and make them interoperable by the use of exchange languages with well-defined semantics [14, 15] such as Hybrid System Interchange Format (HSIF) which gives semantics of hybrids systems in terms of dynamic networks of hybrid automata [16, 17].

Although hybrid automata-based approaches like HSIF can be used to give semantics to block diagram languages [18] and have many advantages (e.g. availability of verification theory for hybrid automata), we consider them not entirely satisfactory from a theoretical standpoint for the following reasons:

- Semantics of a system component (block) is based on the notion of a dynamical system with an internal state, and so the components with the same externally observable behavior can be semantically distinguishable. In our opinion, this is not needed for a high-level semantics which does not intend to describe details of the component's physical/logical implementation.
- Semantics of a system has a computational nature: it describes a sequence of discrete steps, where a step may involve function computations, solving initial-value problems for differential equations (continuous evolution), etc. It may be adequate for certain classes of discrete-continuous systems, but it does not always capture the behavior of a physical realization of a system (and thus may conflict which the view of a system designer).
  For example, a Zeno execution [17] of a hybrid automaton can be described as an infinite sequence of discrete steps which takes a bounded total time (but each step takes a non-zero time). This normally does not correspond to the behavior of a physical system described by the automaton. In many cases this is caused by system modeling simplifications. The conflict is usually resolved by applying a certain method of continuation of an execution beyond Zeno time (regularization, Fillipov solution, etc.) [19]. But an extended execution is not, in fact, a sequence of discrete steps, as it resumes after the accumulation point.
  In our opinion, in the general case, the dynamic behavior of a system should not be restricted to a particular scheme like a sequence of discrete steps and continuous evolutions.

The goal of this paper is to introduce abstract formal models for blocks and block diagrams which overcomes limitations of the classical control/signal-theoretic approach to them and does not enforce implementation details (like internal state-space) or particular kinds of dynamic behavior (like alternation of discrete steps and continuous evolutions) on blocks.

These models can be used to identify the most general properties of block diagram languages which are valid regardless of implementation details. In particular, in the paper we will give a general formulation and conditions for well-definedness of the operation of a system specified by a block diagram for each admissible input signal(s).

To achieve our goal, we will use a *composition-nominative approach* [20]. The main idea of this approach is that semantics of a system is constructed from semantics of components using special operations called compositions, and the syntactic representation of a system reflects this construction.

The paper is organized in the following way:

- In Section 2 we give definitions of the auxiliary notions which are used in the rest of the paper.
- In Section 3 we introduce abstract notions of a block, a connection, a block diagram, and a block composition. We show how they fit into a system design process and give conditions of well-definedness of the operation of a system specified by a block diagram for each admissible input signal(s).

2 Preliminaries

2.1 Notation

We will use the following notation: $\mathbb{N} = \{1, 2, 3, ...\}$, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, $\mathbb{R}_+$ is the set of nonnegative real numbers, $f : A \to B$ is a total function from A to B, $f : A \tilde{\to} B$ is a partial function from A to B, 2^A is the power set of a set A, $f|_X$ is the restriction of a function f to a set X. If A, B are sets, then B^A denotes the set of all total functions from A to B. For a function $f : A \tilde{\to} B$ the symbol $f(x)\downarrow$ ($f(x)\uparrow$) means that $f(x)$ is defined (respectively undefined) on the argument x.

We denote the domain and range of a function as $dom(f) = \{x \,|\, f(x)\downarrow\}$ and $range(f) = \{y \mid \exists x\ f(x)\downarrow \wedge\ y = f(x)\}$ respectively. We will use the the same notation for the domain and range of a binary relation: if $R \subseteq A \times B$, then $dom(R) = \{x \,|\, \exists y\ (x, y) \in R\}$ and $range(R) = \{y \,|\, \exists x\ (x, y) \in R\}$.

We will use the notation $f(x) \cong g(x)$ for the strong equality (where f and g are partial functions): $f(x)\downarrow$ iff $g(x)\downarrow$ and $f(x)\downarrow$ implies $f(x) = g(x)$.

The symbol $\circ$ denotes a functional composition: $(f \circ g)(x) \cong g(f(x))$.

By T we denote the (positive real) time scale $[0, +\infty)$. We assume that T is equipped with a topology induced by the standard topology on $\mathbb{R}$.

Additionally, we define the following class of sets:

$$\mathcal{T}_0 = \{\emptyset, T\} \cup \{[0, x) \,|\, x \in T \backslash \{0\}\} \cup \{[0, x] \,|\, x \in T\}$$

i.e. the set of (possibly empty, bounded or unbounded) intervals with left end 0.

2.2 Multi-valued functions

A multi-valued function [20] assigns one or more resulting values to each argument value. An application of a multi-valued function to an argument is interpreted as a nondeterministic choice of a result.

Definition 1 ([20]). *A (total) multi-valued function from a set A to a set B (denoted as $f : A \xrightarrow{tm} B$) is a function $f : A \to 2^B \setminus \{\emptyset\}$.*

Thus the inclusion $y \in f(x)$ means that y is a possible value of f on x.

2.3 Named sets

We will use a simple notion of a named set to formalize an assignment of values to variable names in program and system semantics.

Definition 2. *([20]) A named set is a partial function $f : V \tilde{\to} W$ from a non-empty set of names V to a set of values W.*

In this definition both names and values are unstructured. A named set can be considered as a partial ("flat") case of a more general notion of nominative data [20] which reflects hierarchical data organizations and naming schemes.

We will use a special notation for the set of named sets: ${}^V W$ denotes the set of all named sets $f : V \tilde{\to} W$ (this notation just emphasises that V is interpreted as a set of names). We consider named sets equal, if their graphs are equal.

An expression of the form $[n_1 \mapsto a_1, n_2 \mapsto a_2, ...]$ (where $n_1, n_2, ...$ are distinct names) denotes a named set d such that the graph of d is $\{(n_1, a_1), (n_2, a_2), ...\}$. A nowhere-defined named set is called an *empty named set* and is denoted as $[]$.

For any named sets d_1, d_2 we write $d_1 \subseteq d_2$ (*named set inclusion*), if the graph of a function d_1 is a subset of the graph of d_2. We extend set-theoretical operations of union $\cup$, intersection $\cap$ and difference $\setminus$ to the partial operations on named sets in the following way: the result of a union (intersection, difference) of named sets (operation's arguments) is a named set d such that the graph of d is the union (intersection, difference) of graphs of the arguments (if such d exists).

3 An Abstract Block Formalism

3.1 Abstract block

Let us introduce abstract notions of a signal as a time-varying quantity and a block as a signal transformer. We will use a real time scale for signals, but we will not require them to be continuous or real-valued. So the signals can be piecewise-constant as well and can be used to represent discrete evolutions.

Informally, a block (see Fig. 1) is a device which receives input signals and produces output signals. We call a collection of input (output) signals an input (resp. output) signal bunch. At each time moment (the value of) a given signal may be present or absent. In the general case, the presence of an input signal at a given time does not imply the presence of an output signal at the same or any other time moment.

A block can operate nondeterministically, i.e. for one input signal bunch it may choose an output signal bunch from a set of possible variants. However, for any input signal bunch there exists at least one corresponding output signal

bunch (although the values of all signals in it may be absent at all times, which means that the block does not produce any output values).

Normally, a block processes the whole input signal bunch, and does or does not produce output values. However, in certain cases a block may not process the whole input signal bunch and may terminate at some time moment before its end. This situation is interpreted as an abnormal termination of a block (e.g. caused by an invalid input).

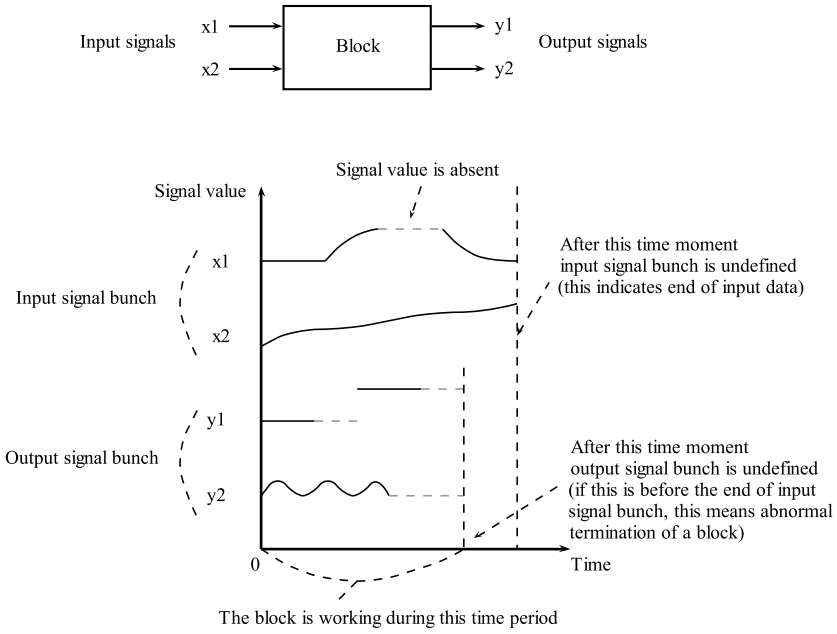

Fig. 1. An illustration of a block with input signals x_1, x_2 and output signals y_1, y_2. The plot displays example evolutions of input and output signals. The input and output signals are lumped into an input and output signal bunch respectively. Solid curves represent (present) signal values. Dashed horizonal segments indicate absence of a signal value. Dashed vertical lines indicate the right boundaries of the domains of signal bunches.

Let us give formal definitions. Let W be a (fixed) non-empty set of values.

Definition 3. *(1) A signal is a partial function from T to W ($f : T \tilde{\to} W$).*
(2) A V-signal bunch (where V is a set of names) is a function $s : T \tilde{\to}^V W$ such that $dom(s) \in \mathcal{T}_0$. The set of all V-signal bunches is denoted as $Sb(V, W)$.
(3) A signal bunch is a V-signal bunch for some V.
(4) A signal bunch s is trivial, if $dom(s) = \emptyset$ and is total, if $dom(s) = T$. A trivial signal bunch is denoted as $\perp$.

(5) For a given signal bunch s, a signal corresponding to a name x is a partial function $t \mapsto s(t)(x)$. This signal is denoted as $s[x]$.
(6) A signal bunch s_1 is a prefix of a signal bunch s_2 (denoted as $s_1 \preceq s_2$), if $s_1 = s_2|_A$ for some $A \in \mathcal{T}_0$.

Note that $\preceq$ on V-signal bunches is a partial order (for an arbitrary V). Later we will need generalized versions of the prefix relation for pairs and indexed families of pairs of signal bunches.

For any signal bunches s_1, s_2, s'_1, s'_2 let us denote $(s_1, s_2) \preceq^2 (s'_1, s'_2)$ iff there exists $A \in \mathcal{T}_0$ such that $s_1 = s'_1|_A$ and $s_2 = s'_2|_A$.

For any indexed families of pairs of signal bunches $(s_j, s'_j)_{j\in J}$ and $(s''_j, s'''_j)_{j\in J}$ of signal bunches let us denote $(s_j, s'_j)_{j\in J} \preceq^{J,2} (s''_j, s'''_j)_{j\in J}$ iff there exists $A \in \mathcal{T}_0$ such that $s_j = s''_j|_A$ and $s'_j = s'''_j|_A$ for all $j \in J$.

It is easy to check that $\preceq^2$ is a partial order on pairs of signal bunches and $\preceq^{J,2}$ is a partial order on J-indexed families of pairs of signal bunches.

A block has a syntactic aspect (e.g. a description in a specification language) and a semantic aspect – a partial multi-valued function on signal bunches.

Definition 4. *(1) A block is an object B (syntactic aspect) together with an associated set of input names $In(B)$, a set of output names $Out(B)$, and a total multi-valued function $Op(B) : Sb(In(B), W) \xrightarrow{tm} Sb(Out(B), W)$ (operation, semantic aspect) such that $o \in Op(B)(i)$ implies $dom(o) \subseteq dom(i)$.*
(2) Two blocks B_1, B_2 are semantically identical, if $In(B_1) = In(B_2)$, $Out(B_1) = Out(B_2)$, and $Op(B_1) = Op(B_2)$.
(3) An I/O pair of a block B is a pair of signal bunches (i, o) such that $o \in Op(B)(i)$. The set of all I/O pairs of B is denoted as $IO(B)$ and is called the input-output (I/O) relation of B.

An inclusion $o \in Op(B)(i)$ means that o is a possible output of a block B on the input i. For each input i there is some output o. The domain of o is a subset of the domain of i. If o becomes undefined at some time t, but i is still defined at t, we interpret this as an error during the operation of the block B (the block cannot resume its operation after t).

Definition 5. *A block B is deterministic, if $Op(B)(i)$ is a singleton set for each $In(B)$-signal bunch i.*

We interpret the operation of a block as a (possibly nondeterministic) choice of an output signal bunch corresponding to a given input signal bunch. However, we would also like to describe this choice as dynamic, i.e. that a block chooses the output signal values at each time t, and in doing so it cannot rely on the future values of the input signals (i.e. values of the input signals at times $t' > t$).

If a block is deterministic, this requirement can be formalized in the same way as the notion of a causal (or nonanticipative) input-output system [26].

Definition 6. *A deterministic block B is causal iff for all signal bunches i_1, i_2 and $A \in \mathcal{T}_0$, $o_1 \in Op(B)(i_1)$, $o_2 \in Op(B)(i_2)$, the equality $i_1|_A = i_2|_A$ implies $o_1|_A = o_2|_A$.*

This means that the value of the output signal bunch at time t can depend only on the values of the input signal at times $\leq t$.

Some works in the domain of systems theory extend the notion of a causal (deterministic) system to nondeterministic systems. However, there is no unified approach to an extension of this kind. For example, in the work [21], a system, considered as a binary relation on (total) signals $S \subseteq A^T \times B^T$, where T is a time domain (Mesarovic time system, [22]) is "non-anticipatory", if it is a union of (graphs of) causal (non-anticipatory) selections from S, i.e, $S = \bigcup\{f : dom(S) \to range(S) \mid f \subseteq S, f \text{ is causal}\}$. In the work [23] the authors define another notion of a "non-anticipatory" or "causal" system in nondeterministic case. In the theory developed in the work [24], the authors use a similar notion of a "precausal" system, which is also defined in [22], as a generalization of the notion of a causal system to the nondeterministic case.

In this work, we generalize the notion of a non-anticipatory system in sense of [23] to blocks and call such blocks *nonanticipative*, and generalize the notion of a non-anticipatory system in sense of [21] to blocks, but call such blocks *strongly nonanticipative*. We will show that strongly nonanticipative block is nonanticipative. We will consider the words "causal" and "nonanticipative" as synonyms when they are used informally, but we will distinguish them in the context of formal definitions to avoid a conflict with Definition 6.

Note, however, that the notion of a strongly nonanticipative block defined below is very different from the notion of a "strictly causal" system, defined in some works [25] as a system which uses only past (but not current or future) values of the input signal(s) to produce a current value of the output signal(s).

Definition 7. *A block B is nonanticipative, if for each $A \in \mathcal{T}_0$ and $i_1, i_2 \in Sb(In(B), W)$, if $i_1|_A = i_2|_A$, then*

$$\{o|_A \mid o \in Op(B)(i_1)\} = \{o|_A \mid o \in Op(B)(i_2)\}.$$

Definition 8. *A block B is a sub-block of a block B' (denoted as $B \trianglelefteq B'$), if $In(B) = In(B')$, $Out(B) = Out(B')$, and $IO(B) \subseteq IO(B')$.*

Informally, a sub-block narrows nondeterminism of a block.

Definition 9. *A block B is strongly nonanticipative, if for each $(i, o) \in IO(B)$ there exists a deterministic causal sub-block $B' \trianglelefteq B$ such that $(i, o) \in IO(B')$.*

Informally, the operation of a strongly nonanticipative block B can be interpreted as a two-step process:

1. before receiving the input signals, the block B (nondeterministically) chooses a deterministic causal sub-block $B' \trianglelefteq B$ (response strategy);
2. the block B' receives input signals of B and produces the corresponding output signals (response) which become the output signals of B.

Intuitively, it is clear that in this scheme at any time the block B does not need a knowledge of the future of its input signals in order produce the corresponding output signals.

Lemma 1. *If B is a deterministic block, then B is causal iff B is nonanticipative.*

Proof. Follows immediately from Definition 6.

The following theorem gives a characterization of a nonanticipative block which does not rely on comparison of sets of signal bunches.

Theorem 1. *A block B is nonanticipative iff the following holds:*

(1) if $(i,o) \in IO(B)$ and $(i',o') \preceq^2 (i,o)$, then $(i',o') \in IO(B)$;
(2) if $o \in Op(B)(i)$ and $i \preceq i'$, then $(i,o) \preceq^2 (i',o')$ for some $o' \in Op(B)(i')$.

Proof. (1) Assume that (1) and (2) are satisfied. Assume that $A \in \mathcal{T}_0$, $i_1, i_2 \in Sb(In(B), W)$, and $i_1|_A = i_2|_A$. Let $o \in Op(B)(i_1)$. Then from assumption (1) we have $o|_A \in Op(B)(i_1|_A)$, because $(i_1|_A, o|_A) \preceq^2 (i_1, o)$. Moreover, $i_1|_A \preceq i_2$, because $i_1|_A = i_2|_A$. Thus $(i_1|_A, o|_A) \preceq^2 (i_2, o')$ for some $o' \in Op(B)(i_2)$ by assumption (2). It is not difficult to check that $o|_A \in \{o''|_A \mid o'' \in Op(B)(i_2)\}$. Because i_1, i_2, A are arbitrary, B is nonanticipative by Definition 7.

(2) Assume that B is nonanticipative. Let us prove (1). Assume that $(i,o) \in IO(B)$ and $(i',o') \preceq^2 (i,o)$. Then $i' = i|_A$ and $o' = o|_A$ for some $A \in \mathcal{T}_0$. Then $i'|_A = (i|_A)|_A = i|_A$, whence

$$o' = o|_A \in \{o''|_A \mid o'' \in Op(B)(i)\} = \{o''|_A \mid o'' \in Op(B)(i')\}$$

by Definition 7. Then $o' = o''|_A$ for some $o'' \in Op(B)(i')$. Moreover, $dom(o'') \subseteq dom(i') \subseteq A$. Thus $o' = o''$ and $(i',o') \in IO(B)$.
Let us prove (2). Assume that $o \in Op(B)(i)$ and $i \preceq i'$. Then $i = i'|_A$ for some $A \in \mathcal{T}_0$. Then $i|_A = (i'|_A)|_A = i'|_A$, whence

$$o|_A \in \{o''|_A \mid o'' \in Op(B)(i)\} = \{o''|_A \mid o'' \in Op(B)(i')\}$$

by Definition 7. Then $o|_A = o'|_A$ for some $o' \in Op(B)(i')$. Moveover, $dom(o) \subseteq dom(i) \subseteq A$, whence $o = o|_A = o'|_A$. Thus $(i,o) \preceq^2 (i',o')$.

Theorem 2. *(About strongly nonanticipative block)*

(1) If a block B is strongly nonanticipative, then it is nonanticipative.
(2) There exists a nonanticipative block which is not strongly nonanticipative.

Proof (Sketch).

(1) Assume that B is strongly nonanticipative. Let $\mathcal{R}$ be the set of all relations $R \subseteq IO(B)$ such that R is an I/O relation of a nonanticipative block. For each $R \in IO$ let us define a block B_R such that $IO(B_R) = R$, $In(B_R) = In(B)$, $Out(B_R) = Out(B)$. Let $\mathcal{B} = \{B_R \mid R \in \mathcal{R}\}$. Then each element of $\mathcal{B}$ is nonanticipative. From Definition 9 and Lemma 1 we have $IO(B) \subseteq \bigcup \mathcal{R} = \bigcup_{B' \in \mathcal{B}} IO(B')$. On the other hand, $IO(B') \subseteq IO(B)$ for any $B' \in \mathcal{R}$, so $IO(B) = \bigcup_{B' \in \mathcal{B}} IO(B')$. It is easy to see from Theorem 1 that (nonempty) union of I/O relations of nonanticipative block is an I/O relation of a nonanticipative block. Thus B is nonanticipative.

(2) Assume that $W = \mathbb{R}$. Let $f : \mathbb{R} \to \mathbb{R}$ be a function that is discontinuous at some point (e.g. a signum function). Let use define a block B such that $In(B) = \{x\}$ and $Out(B) = \{y\}$ for some names x, y, and for each $i \in Sb(In(B), \mathbb{R})$, $Op(B)(i)$ is defined as follows:

- if $dom(i[x]) = T$ and $\lim_{t\to+\infty} i[x](t)$ exists and finite, then $Op(B)(i)$ is the set of all $\{y\}$-signal bunches o such that $dom(o) = dom(o[y]) = T$ and
$$\lim_{t\to+\infty} o[y](t) = f\left(\lim_{t\to+\infty} i[x](t)\right);$$
- otherwise, $Op(B)(i)$ is the set of all $\{y\}$-signal bunches o such that $dom(o) = dom(o[y]) = \bigcup\{A \in \mathcal{T}_0 \mid A \subseteq dom(i[x])\}$.

Obviously, in this definition $Op(B)(i) \neq \emptyset$ (because $\mathcal{T}_0$ is closed under unions) and $dom(o) \subseteq dom(i)$ for each $o \in Op(B)(i)$. So B is indeed a block. Using Theorem 1 it is not difficult to show that B is nonanticipative. Suppose that B has a a deterministic causal sub-block B'. Let $a \in \mathbb{R}$ and $a_k \in \mathbb{R}$, $k = 1, 2, ...$ be a sequence such that $\lim_{k\to\infty} a_k = a$. Let us show that $\lim_{k\to\infty} f(a_k) = f(a)$. Let us define sequences $i_k \in Sb(\{y\}, \mathbb{R})$, $o_k \in Sb(\{y\}, \mathbb{R})$, and $t_k \in T$, $k = 1, 2, ...$ by induction as follows.
Let $i_1(t) = [x \mapsto a_1]$ for all $t \in T$, o_1 be a unique member of $Op(B')(i_1)$, and $t_1 = 0$. If $i_1, i_2, ..., i_k$ are already defined, let $i_{k+1}(t) = i_k(t)$, if $t \in [0, t_k]$ and $i_{k+1}(t) = [x \mapsto a_{k+1}]$, if $t \in T\backslash[0, t_k]$. Let o_{k+1} be a unique member of $Op(B')(i_{k+1})$. Because $B' \trianglelefteq B$, $dom(o_{k+1}) = dom(o_{k+1}[y]) = T$ and $\lim_{t\to+\infty} o_{k+1}[y](t) = f(\lim_{t\to+\infty} i_{k+1}[x](t)) = f(a_{k+1})$. Then let

$$t_{k+1} = 1 + \max\{t_k, \inf\{\tau \in T \mid$$

$$\sup\{|o_{k+1}[y](t) - f(a_{k+1})| \mid t \geq \tau\} \leq \frac{1}{k+1}\}\}$$

We have defined sequences i_k, o_k, t_k. The sequence t_k, $k = 1, 2, ...$ is a strictly increasing and unbounded from above and $t_1 = 0$.
Let i be a $\{x\}$-signal bunch such that $dom(i) = T$, $i(t_1) = i_1(t_1)$, and $i(t) = i_{k+1}(t)$, if $t \in (t_k, t_{k+1}]$, $k \in \mathbb{N}$, and o be a (unique) member of $Op(B')(i)$. We have $i_{k+1}[x](t) = a_{k+1}$ for all $k = 1, 2, ...$ and $t > t_k$. Then $i[x](t) \in \{a_{k+1}, a_{k+2}, ...\}$ for all $k \in \mathbb{N}$ and $t > t_k$. For each $\epsilon > 0$ there exists $k \in \mathbb{N}$ such that $|a_{k'} - a| < \epsilon$ for all $k' \geq k$, whence $|i[x](t) - a| < \epsilon$ for all $t > t_k$. Thus $\lim_{t\to+\infty} i[x](t) = a$. Then $dom(o) = dom(o[y]) = T$ and $\lim_{t\to+\infty} o[y](t) = f(a)$, because $B' \trianglelefteq B$.
On the other hand, $i_{k+1}|_{[0,t_k]} = i_k|_{[0,t_k]}$ for all $k \in \mathbb{N}$. Because t_k is an increasing sequence, we have $i_{k'}|_{[0,t_k]} = i_k|_{[0,t_k]}$ for all k and $k' \geq k$. Besides, $i|_{(t_k,t_{k+1}]} = i_{k+1}|_{(t_k,t_{k+1}]}$ for all $k \in \mathbb{N}$, whence $i|_{(t_k,t_{k+1}]} = i_{k'}|_{(t_k,t_{k+1}]}$ for all $k' \geq k + 1$. Also, $i_k(t_1) = i_1(t_1)$ for all $k \in \mathbb{N}$. Then $i|_{[0,t_k]} = i|_{\{t_1\}\cup(t_1,t_2]\cup...\cup(t_{k-1},t_k]} = i_k|_{[0,t_k]}$ for all $k = 2, 3, ...$, whence $o|_{[0,t_k]} = o_k|_{[0,t_k]}$, because B' is causal. Then $o(t_k) = o_k(t_k)$ for all $k = 2, 3, ...$, and from the definition of t_k we have $|o[y](t_k) - f(a_k)| = |o_k[y](t_k) - f(a_k)| \leq \frac{1}{k}$ for all $k = 2, 3, ...$ This implies that $\lim_{k\to\infty} f(a_k) = f(a)$, because $\lim_{t\to+\infty} o[y](t) = f(a)$. We conclude that f is sequentially continuous and thus is continuous.

This contradicts our choice of f as a discontinuous function. Thus B has not deterministic causal sub-blocks. Consequently, B is not strongly nonanticipative, though it is nonanticipative.

The proof of this theorem gives a reason of why Definition 9 better captures a intuitive idea of causality than Definition 7. Consider, for example, the block B constructed in the proof of the item (2) of Theorem 2, when f is the signum function (i.e., $f(0) = 0$, $f(x) = 1$, if $x > 0$, and $f(x) < 0$, if $x < 0$). Then B outputs a signal which converges to 1 (as $t \to +\infty$) whenever the input signal converges to a positive number (as $t \to +\infty$). Moreover, it outputs a signal which converges to 0 whenever the input signal converges to 0. This implies that when the block receives a decreasing positive input signal which tends to 0, it decides to output values which are close to 0 starting from some time t. Intuitively, after reading the input signal until time t, the block decides that 0 is a more likely limit of the input signal than a positive value, but such a decision cannot be based on the past values of the input signal, so it requires some knowledge of the future of the input signal. These informal observations are captured by the fact that B has no deterministic causal sub-blocks.

In the rest of the paper we will focus on strongly nonanticipative blocks, as more adequate models of (real-time) information processing systems.

Consider an example of a strongly nonanticipative block.

Let u, y be names. Assume that $W = \mathbb{R}$.

Example 1. Let B be a block such that $In(B) = \{u\}$, $Out(B) = \{y\}$, and for each i, $Op(B)(i) = \{o_1(i), o_2(i)\}$, where $o_1(i), o_2(i) \in Sb(Out(B), W)$ are signal bunches such that

- $dom(o_1(i)) = dom(o_2(i)) = dom(i)$;
- $o_1(i)(t) = [y \mapsto i[u](t)]$ for all $t \in dom(i)$;
- $o_2(i)(t) = [y \mapsto 2i[u](t)]$ for all $t \in dom(i)$.

Informally, this means that B is a gain block with a slope which is either 1 or 2 during the whole duration of the block's operation.

Obviously, B satisfies Definition 4(1). Let us check that it is strongly nonanticipative. For $j = 1, 2$ let $B_j \trianglelefteq B$ be a sub-block such that $Op(B_j)(i) = \{o_j(i)\}$ for all $i \in Sb(In(B), W)$ (i.e. B_1 always selects $o_1(i)$ from $Op(B)(i)$ and B_2 always selects $o_2(i)$).

The blocks B_1, B_2 are deterministic and it is easy to see that they are causal. Obviously, each I/O pair $(i, o) \in IO(B)$ belongs either to $IO(B_1)$, or to $IO(B_2)$, so B is strongly nonanticipative.

Now let us consider an example of a block which is not nonanticipative.

Example 2. Let B' be a block such that $In(B') = \{u\}$, $Out(B') = \{y\}$, and

- $Op(B')(i) = \{o_1\}$, where $dom(o_1) = dom(i)$ and $o_1(t) = [y \mapsto 1]$ for all $t \in dom(i)$, if $dom(i[u]) = T$;

– $Op(B')(i) = \{o_2\}$, where $dom(o_2) = dom(i)$ and $o_2(t) = [y \mapsto 0]$ for all $t \in dom(i)$, otherwise.

Informally, the block B' decides whether its input signal u is total. It is easy to see that B' indeed satisfies Definition 4(1), but the condition (1) of Theorem 1 is not satisfied, because $(i, o) \in IO(B')$, where $i(t) = [u \mapsto 0]$ for all $t \in T$, $o(t) = [y \mapsto 1]$ for all $t \in T$, and $(i|_{[0,1]}, o|_{[0,1]}) \preceq^2 (i, o)$, but $(i|_{[0,1]}, o|_{[0,1]}) \notin IO(B')$. So B' is not nonanticipative. Informally, the reason is that at each time t the current value of y depends on the entire input signal.

3.2 Composition of blocks

By connecting inputs and outputs of several (strongly nonanticipative) blocks one can form a larger block – a composition of blocks (see Fig. 2). We assume that an output can be connected to several inputs, but each input can be connected to no more than one output. Unconnected inputs and outputs of constituent blocks become inputs and output of the composition. Connections are interpreted as signal equality constraints and they always relate an output of some block ("source") with an input of the same or another block ("target"). We represent connections in the graphical form (like in Fig. 2) as arrows connecting blocks.

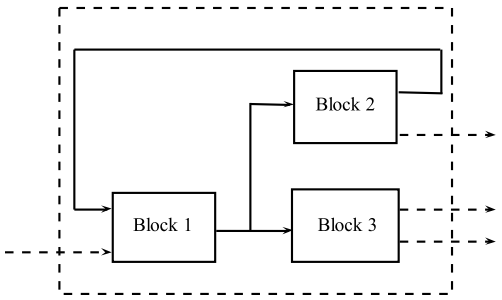

Fig. 2. An informal illustration of a block composition. Three blocks are composed to form a larger block (a dashed rectangle). Solid arrows denote connections between blocks. Dashed arrows denote unconnected inputs/outputs of the blocks 1, 2, 3 (which become the input/outputs of the dashed block).

Definition 10. *(1) A block diagram is a pair $((B_j)_{j\in J}, \rightarrowtail)$ of an indexed family of blocks $(B_j)_{j\in J}$ and an injective binary relation $\rightarrowtail \subseteq V_{out} \times V_{in}$, which is called an interconnection relation, where*

$$V_{in} = \bigcup_{j\in J} \{j\} \times In(B_j), V_{out} = \bigcup_{j\in J} \{j\} \times Out(B_j).$$

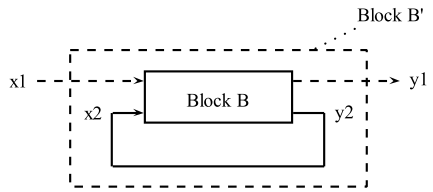

Fig. 3. A strongly nonanticipative block B is composed to obtain a block B' (a dashed rectangle). A solid loop arrow denotes a connection between an output and an input of B. Dashed arrows denote unconnected inputs and outputs of B which become the inputs and outputs of B'.

(2) A block diagram $((B_j)_{j\in J}, \rightarrowtail)$ *is called regular, if each* B_j, $j \in J$ *is strongly nonanticipative.*

Note that a diagram may consist of an infinite set of blocks. A relation $(j, x) \rightarrowtail (j', x')$ means that the output x of the j-th block is connected to the input x' of the j'-th block.

A block diagram is only a syntactic aspect of a block composition. We define semantics of a block composition only for strongly nonanticipative blocks.

To describe it informally consider Fig. 3. The connection between $y2$ and $x2$ means a signal equality constraint. The block B chooses (nondeterministically) some deterministic sub-block $B_0 \trianglelefteq B$. When a signal starts flowing into the input $x1$, the block B' tries to choose the initial values for the signals of $y1$, $y2$, $x2$ so that they satisfy the operation of the block B_0 ($Op(B_0)$) and the signals of $x2$ and $y2$ have the same values. If such initial values do not exist, the block B' terminates (the output signal bunch is nowhere defined). Otherwise, B' continues to operate in the similar way until either the signals of $y1$, $y2$, $x2$ cannot be continued, or the input signal ($x1$) ends.

Definition 11. *Let* $((B_j)_{j\in J}, \rightarrowtail)$ *be a regular block diagram.*
A block B *is a composition of* $(B_j)_{j\in J}$ *under the interconnection relation* $\rightarrowtail$*, if*

- $In(B) = (\bigcup_{j\in J}\{j\} \times In(B_j))\backslash range(\rightarrowtail)$,
- $Out(B) = (\bigcup_{j\in J}\{j\} \times Out(B_j))\backslash dom(\rightarrowtail)$,
- $Op(B)(i)$ *is the set of all* $Out(B)$*-signal bunches* o *such that there exist deterministic causal sub-blocks* $B'_j \trianglelefteq B_j$, $j \in J$ *and an indexed family* $(i_j, o_j)_{j\in J} \in X_m(i)$ *such that*
  (1) $dom(o) = dom(o_j)$ *for all* $j \in J$,
  (2) $o[(j, x)] = o_j[x]$ *for all* $(j, x) \in Out(B)$,
  where $X_m(i)$ *is the set of* $\preceq^{J,2}$*-maximal elements of* $X(i)$*, and* $X(i)$ *is the set of all indexed families of pairs of signal bunches* $u = (i_j, o_j)_{j\in J}$ *such that*
  (3) $dom(i_j) = dom(o_j) = dom(i_{j'}) = dom(o_{j'}) \subseteq dom(i)$ *for all* $j, j' \in J$,
  (4) $i_j[x] = i|_{dom(i_j)}[(j, x)]$ *for each* $(j, x) \in In(B)$,
  (5) $(i_j, o_j) \in IO(B'_j)$ *for each* $j \in J$,

(6) $(j,x) \rightarrowtail (j',x')$ *implies* $o_j[x] = i_{j'}[x']$.

In this definition, i_j and o_j denote the input and output signal bunches of the j-th block. The set $X_m(i)$ contains maximally extended (in sense of the relation $\preceq^{J,2}$) indexed families of signal bunches defined on a subset of the domain of i (the input signal bunch of B) which satisfy constraints imposed by the interconnection relation. Any such family gives a possible output of B for the given i by the condition (2), i.e. output signals of B are obtained from the output signals of the sub-blocks B'_j.

It is clear that any two compositions of $(B_j)_{j\in J}$ under $\rightarrowtail$ are semantically identical.

Lemma 2. *(Continuity of the operation of a causal deterministic causal block). Let B be a deterministic causal block. Let $c \subseteq Sb(In(B), W)$ be a non-empty $\preceq$-chain, i^* be its supremum (in sense of $\preceq$), and $o^* \in Op(B)(i^*)$. Then o^* is a supremum of $\bigcup_{i\in c} Op(B)(i)$ (in sense of $\preceq$).*

Proof. Follows from Definition 6.

Theorem 3. *Let $((B_j)_{j\in J}, \rightarrowtail)$ be a regular block diagram. Then*

(1) A composition of $(B_j)_{j\in J}$ under $\rightarrowtail$ exists.
(2) If B is a composition of $(B_j)_{j\in J}$ under $\rightarrowtail$, then B is strongly nonanticipative.

The proof follows from Lemma 2 and Definition 11.

3.3 Specification and implementation

Above we have considered a block as an abstract model of a real component. However, it can also be considered as a specification of requirements for a component. Let B^{spec}, B^{impl} be two strongly nonanticipative blocks. Let us call them a specification block and implementation block respectively.

Definition 12. *B^{impl} is a refinement of B^{spec}, if B^{impl} is a sub-block of B^{spec}.*

I.e. an implementation should have the the same input and output names as a specification, and for each input, an output of an implementation should be one of the possible outputs of a specification. We generalize this to diagrams.

Let $D = ((B_j)_{j\in J}, \rightarrowtail)$ and $D' = ((B'_j)_{j\in J'}, \rightarrowtail')$ be regular block diagrams.

Definition 13. *D is a refinement of D', if $J = J'$, B_j is a refinement of B'_j for each $j \in J$, and the relations $\rightarrowtail$ and $\rightarrowtail'$ coincide.*

Theorem 4. *(Compositional refinement) Let B be a composition of $(B_j)_{j\in J}$ under $\rightarrowtail$ and B' be a composition of $(B'_j)_{j\in J'}$ under $\rightarrowtail'$. If D is a refinement of D', then B is a refinement of B'.*

The proof follows from Definition 9, 11, and transitivity of the sub-block relation.

This theorem can be considered as a foundation of a modular approach [29, 30] to system design:

1. Create specifications of the system components (B'_j, $j \in J'$) and connect them ($\rightarrowtail'$), as if they were real components.
2. Analyze a composition of specifications (B') to ensure that any of its implementations ($B'' \trianglelefteq B'$) satisfies requirements to the final system.
3. Create an implementation (B_j) for each specification (B'_j).
4. Connect implementations (according to $\rightarrowtail'$). Then the composition of implementations (B) is a final system which satisfies design requirements.

We consider the steps 1 and 3 domain- and application-specific. The conclusion of the step 4 is addressed in Theorem 4. Step 2 requires some verification method which depends on the nature of requirements.

One of the most basic and common requirements is that the operation of B' is defined on all input signal bunches which are possible in the context of a specific application of this composition. This trivially holds because of Theorem 3 and our definition of a block. However, B' may terminate abnormally on some or all input signal bunches of interest (as we have noted, we interpret the situation when $o \in Op(\tilde{B})(i)$ and $dom(o) \subset dom(i)$ for some block $\tilde{B}$ as abnormal termination of $\tilde{B}$ on i). So the requirement can be reformulated as follows: B' never terminates abnormally on any input signal bunch from a given set IN (this implies that the same property holds for B). We will call this property as well-definedness of the operation of B' on IN and study it in the next subsection.

3.4 Well-definedness of the operation of a composition of blocks

Let B be a block and IN be some set of $In(B)$-signal bunches.

Definition 14. *The operation of B is well-defined on IN, if $dom(i) = dom(o)$ for each $i \in IN$ and $o \in Op(B)(i)$.*

Let $D = ((B_j)_{j\in J}, \rightarrowtail)$ be a regular block diagram and B be a composition of $(B_j)_{j\in J}$ under $\rightarrowtail$. Let $\mathcal{F}$ be the set of all families of blocks of the form $(B'_j)_{j\in J}$, where for each $j \in J$, B'_j is a deterministic causal sub-block of B_j.

For each $In(B)$-signal bunch i and a family of blocks $F = (B'_j)_{j\in J}$ let $X^F(i)$ be the set of all indexed families of pairs of signal bunches $u = (i_j, o_j)_{j\in J}$ which satisfy conditions (3)-(6) of Definition 11 (for B and $(B'_j)_{j\in J}$).

Let $X^F_m(i)$ denote the set of all $\preceq^{J,2}$-maximal elements of $X^F(i)$. For any indexed family of signal bunches $u = (i_j, o_j)_{j\in J}$ let $O(u)$ denote the set of all $Out(B)$-signal bunches o which satisfy conditions (1)-(2) of Definition 11.

For any $u = (i_j, o_j)_{j\in J} \in X^F(i)$, the domains of i_j, o_j for all $j \in J$ coincide. Denote by $cdom(u)$ this common domain (we assume $cdom(u) = T$, if $J = \emptyset$).

From Definition 11 we have $Op(B)(i) = \bigcup_{F\in\mathcal{F}} \bigcup_{u\in X^F_m(i)} O(u)$ for each i. Then from Definition 14 we get the following simple criterion:

Theorem 5. *The operation of B is well-defined on IN iff for each $i \in IN$, $F \in \mathcal{F}$, and $u \in X^F(i)$, if $cdom(u) \subset dom(i)$, then $u \notin X^F_m(i)$.*

This criterion means that B is well-defined, if each $u \in X^F(i)$, the common domain of which does not cover $dom(i)$, is extendable to a larger $u' \in X^F(i)$ (in sense of $\preceq^{J,2}$). We will call it a local extensibility criterion, because, basically, to prove well-definedness, we only need to show that the members of the family u can be continued onto a time segment $[0, \sup cdom(u) + \epsilon]$ for some small $\epsilon > 0$ (under constraints imposed by the interconnection relation $\rightarrowtail$). Locality is especially useful when a block diagram contains "delay" blocks (possibly working as variable delays), because constraints imposed by connections between blocks reduce over small time intervals.

A drawback of this criterion is that it requires checking local extensibility of signal bunches satisfying the I/O relations ($IO(B'_j)$) of arbitrarily chosen deterministic causal sub-blocks $B'_j \trianglelefteq B_j$ (condition (5) of Definition 11), which are not explicitly expressed in terms of I/O relations of B_j, $j \in J$.

For this reason, we seek for a condition of well-definedness in terms of I/O relations of the constituents of the composition ($IO(B_j), j \in J$).

Let $X(i)$ denote the set $X^F(i)$, where $F = (B_j)_{j \in J}$. Note that F may not be a member of $\mathcal{F}$.

Theorem 6. *The operation of B is well-defined on IN iff for each $i \in IN$ and $u \in X(i)$ there exists $u' \in X(i)$ such that $u \preceq^{J,2} u'$ and $cdom(u') = dom(i)$.*

The proof follows from Theorem 5 and Definition 9.

References

1. Simulink - Simulation and Model-Based Design, `http://www.mathworks.com/products/simulink`
2. Campbell, S.L., Chancelier, J.-P., Nikoukhah, R.: Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4. Springer (2010)
3. Multi-Engineering Modeling and Simulation – Dymola, `http://www.3ds.com/products/catia/portfolio/dymola`
4. SCADE Suite, `http://www.esterel-technologies.com/products/scade-suite`
5. Caspi, P., Pilaud, D., Halbwachs, N., Plaice, J.: LUSTRE: A declarative language for programming synchronous systems. In: 14th Annual ACM Symp. on Principles of Programming Languages, Munich, Germany, pp. 178-188 (1987)
6. Henzinger, T., Horowitz, B., Kirsch, C.: Giotto: A Time-Triggered Language for Embedded Programming. First International Workshop on Embedded Software, EMSOFT'01, pp. 166-184 (2001)
7. Lublinerman, R., Tripakis, S.: Modular Code Generation from Triggered and Timed Block Diagrams. In: IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 147-158 (2008)
8. Sontag, E.D.: Mathematical Control Theory: Deterministic Finite Dimensional Systems. Second Edition, Springer, New York (1998)
9. Proakis, J., Manolakis, D.: Digital Signal Processing: Principles, Algorithms and Applications, 4th ed. Pearson (2006)
10. Tiwari, A.: Formal semantics and analysis methods for Simulink Stateflow models. Unpublished report, SRI International (2002)

11. Bouissou, O., Chapoutot, A.: An operational semantics for Simulink's simulation engine. LCTES 2012, pp. 129-138. (2012)
12. Agrawal, A., Simon, G., Karsai, G.: Semantic translation of Simulink/Stateflow models to hybrid automata using graph transformations. Electronic Notes in Theoretical Computer Science 109, 43-56 (2004)
13. Marian, N., Ma, Y.: Translation of Simulink Models to Component-based Software Models. In: 8-th Int. Workshop on Research and Education in Mechatronics, 14-15 June 2007, Talin University of Technology, Estonia (2007)
14. Pinto, R., Sangiovanni-Vincentelli, A., Carloni L.P., Passerone, R.: Interchange formats for hybrid systems: Review and proposal. In: HSCC 05: Hybrid Systems Computation and Control. Springer-Verlag, pp. 526–541 (2005)
15. Beek, D.A., Reniers, M.A., Schiffelers, R.R., Rooda, J. E.: Foundations of a Compositional Interchange Format for Hybrid Systems. In: HSCC'07, pp. 587-600 (2007)
16. Henzinger, T.: The theory of hybrid automata. In: IEEE Symposium on Logic in Computer Science, pp. 278–292 (1996)
17. Goebel, R., Sanfelice, R., Teel, R.: Hybrid dynamical systems. In: IEEE Control Systems Magazine 29, 29–93 (2009)
18. Schrammel, P., Jeannet, B.: From hybrid data-flow languages to hybrid automata: a complete translation. In: HSCC 2012: pp. 167–176 (2012)
19. Camhbel, M.,K., Heemels, A.J., van der Schaft, A.J., Schumacher, J.M.: Solution concepts for hybrid dynamical systems. In: Proc. IFAC 15th Triennial World Congress, Barcelona, Spain (2002)
20. Nikitchenko, N.S.: A composition nominative approach to program semantics. Technical report IT-TR 1998-020, Technical University of Denmark, 103 p. (1998)
21. Windeknecht, T.G.: Mathematical systems theory: Causality. Mathematical systems theory 1, pp. 279-288 (1967)
22. Mesarovic, M.,D., Takahara, Y.: Abstract systems theory. Springer, Berlin Heidelberg New York, 439 p. (1989)
23. Foo, N., Peppas, P.: Realization for Causal Nondeterministic Input-Output Systems. Studia Logica 67, pp. 419-437 (2001)
24. Lin, Y.: General systems theory: A mathematical approach. Springer, 382 p. (1999)
25. Matsikoudis, E., Lee, E.: On Fixed Points of Strictly Causal Functions. Technical report UCB/EECS-2013-27, EECS Department, University of California, Berkeley (2013).
26. Williems, J.: Paradigms and puzzles in the theory of dynamical systems. In: IEEE Transactions on Automatic Control 36, pp. 259-294 (1991)
27. Williems, J.: On Interconnections, Control, and Feedback. In: IEEE Transactions on Automatic Control 42, pp. 326-339 (1997)
28. Williems, J.: The behavioral approach to open and interconnected systems. In: IEEE Control Systems Magazine, pp. 46-99 (2007)
29. Baldwin, C.Y., Clark, K.B.: Design Rules, Volume 1: The Power of Modularity. MIT Press (2000)
30. Tripakis, S., Lickly, B., Henzinger, T., Lee, E.: On relational interfaces. Proceedings of EMSOFT'2009, pp. 67-76 (2009)
31. Ivanov, Ie.: A criterion for global-in-time existence of trajectories of nondeterministic Markovian systems. Communications in Computer and Information Science 347, pp. 111-130, Springer (2012)
32. Carloni, L.P., Passerone, R., Pinto A.: Languages and Tools for Hybrid Systems Design. Foundations and Trends in Design Automation 1, 1-204 (2006)