Network Delay-Insensitivity and Udding's Rules*

H. ZHANG
Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4

J. A. BRZOZOWSKI
Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Abstract

We begin with a network N of components connected by delay-free wires. We analyze the behavior of N, assuming arbitrary component delays, and take this behavior to be correct. We then study the effects of wire delays, and consider N to be a *delay-insensitive network* if its behavior remains correct with arbitrary wire delays.

It is known that network delay-insensitivity is equivalent to a generalized form of semi-modularity, as long as we associate delays with both components and wires. We prove in this paper that, under this assumption, the semi-modularity of a network's behavior implies the delay-insensitivity of its components, in the sense that each component's trace structure satisfies Udding's rules, when the component operates within the network. Our results have the effect of unifying three concepts: Muller's semi-modularity, our notion of network delay-insensitivity, and Udding's rules for delay-insensitive components.

1 Introduction

In contrast to a synchronous circuit, whose operation is under the control of a global clock signal, an asynchronous circuit uses local handshaking among its components to implement sequencing. Potential advantages in using asynchronous circuits include lower energy consumption, avoidance of clock distribution problems, and higher speed (average-case instead of worst-case performance).

An asynchronous circuit is said to be delay-insensitive if its functional correctness is independent of the delays in its components and in the wires connecting the components. Delay-insensitive circuits have received special attention [1, 2, 4, 5, 9, 10, 12, 13, 14, 15, 16]. In such circuits, it is possible to replace some components by faster or slower ones, or change the material used for connection wires, without affecting the correctness of the circuit operation, thus making circuit design more flexible. Another important reason for designing delay-insensitive circuits comes from the fact that wire delays do not scale down proportionally to the other factors affecting the behavior of a circuit, such as switching time and clock period, when the size of that circuit is scaled down [11]. Moreover, the metastability phenomenon [3] becomes a manageable problem in delay-insensitive circuits.

Early work on circuit behaviors in the presence of arbitrary delays traces back to D. E. Muller's theory of "speed-independent" circuits [7, 10], which operate correctly in spite of arbitrary component delays (wire delays are considered negligible).

*This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871, and was done when the first author was at the University of Waterloo.

C. E. Molnar et al. introduced the "foam-rubber wrapper" postulate [9] to help describe delay-insensitive specifications of circuit components. A component is viewed as being wrapped by a "foam-rubber wrapper." It sends signals to and receives signals from the inner surface of the wrapper, whereas its environment does so through the outer surface of the wrapper. Signals are transmitted across the wrapper, whose shape assumes no specific form and may change over time; this corresponds to arbitrary delays in the wires connecting the component to its environment.

Following Molnar's informal idea, J. T. Udding gave the first formal definition (based on trace theory) of delay-insensitivity [13, 14]. He suggested that a specification, in the form of a *trace structure*, is delay-insensitive if there is no *computation interference* or *transmission interference*. He established several rules which define the class of delay-insensitive specifications; we refer to these rules as the *JTU-Rules*. A more detailed description of trace theory and Udding's work is given in Section 5. Other definitions of delay-insensitivity can be found in [4, 5].

Almost all the definitions of delay-insensitivity in the existing literature deal mainly with specifications of asynchronous circuit components using trace structures [4, 14, 15] or regular expression-like programs [5], which represent all possible communication behaviors between the component and its environment. And, the approach in defining delay-insensitivity has always begun by examining the behavior of an individual component. In this paper, we focus on delay-insensitivity of asynchronous *networks* whose components can be any sequential machines of the Moore type. We define network delay-insensitivity using the notion of behavioral equivalence in the presence of arbitrary delays. A network N is delay-insensitive if the behavior of any network $\hat{N}$ derived from N by the addition of delays is weakly bisimilar [8] to the behavior of N, where transitions occurring on the added delays are considered unobservable. N. Shintel and M. Yoeli [12] have used a similar delay-addition scheme in their definition of delay-insensitive realization.

We relate the delay-insensitivity of a network to the delay-insensitive behaviors of its components, via the concept of semi-modularity of behaviors. Semi-modularity of circuit behaviors was initially studied by Muller [7, 10]. Roughly speaking, semi-modularity requires that an enabled action cannot be disabled by another action. In this paper, we use a more general definition of semi-modularity [16] to reflect nondeterminism in network behaviors. We show that for a special class of networks, the delay-dense networks, the semi-modularity of a network's behavior implies the delay-insensitivity of its components, in the sense that each component's trace structure satisfies Udding's delay-insensitivity rules, where a network is delay-dense if, of each pair of adjacent network components, at least one is a delay element.

The paper is organized as follows. In Sections 2–4, we describe our network model and our notion of network delay-insensitivity and semi-modularity in a somewhat informal manner. Section 5 gives Udding's delay-insensitivity rules. In Section 6, we present the main theorem of paper. A proof sketch of the theorem is provided in Section 7.

2 Network Model

2.1 Modules

We model the components of an asynchronous circuit by "modules." A *module* M is characterized by a single *internal state variable* y, which takes on values from the state set $\mathcal{S}$ of M; M has m binary inputs $x_1, \ldots, x_m$ and p binary outputs $z_1, \ldots, z_p$; the *excitation function* δ of M maps the current input state and the current internal state of M to an nonempty *set* of possible next states; the *output function* λ maps the internal state of M to its output state.

Example 2.1 A DELAY module models a wire with delay. It has one input, one output, and its associated functions are defined in Table 1. This module is deterministic in the sense that each excitation contains a single state. □

Table 1: Functions of a DELAY module.

y	$x = 0$	$x = 1$	$\lambda(y)$
0	$\{0\}$	$\{1\}$	0
1	$\{0\}$	$\{1\}$	1
	$\delta(x,y)$		

Let a be the current input state of M and b the current internal state. The pair $t = (a, b)$ represents a total state of M. The set $T = \delta(t) \subseteq \mathcal{S}$ is the *excitation* of M in state t. We require that either $T = \{b\}$ or $b \notin T$. If $T = \{b\}$, then the total state t and the module are said to be *stable*, and the internal state cannot change. If $T \neq \{b\}$, then the total state and the module are *unstable* and the internal state may change to any state $b' \in T$ at any time. The state b' is nondeterministically selected by the module. The distinction between the current and the next state of a module allows us to associated a delay with the module. However, if the internal state changes, and the outputs computed by the output function λ differ from the outputs before the change, the new output values appear *instantaneously* together with the internal state change. An example of a nondeterministic module is shown below.

Example 2.2 An ARBITER module grants requests from either of its two inputs. It is nondeterministic: When both inputs are 1 and the module is in state 0, the next state is arbitrarily chosen from the set $\{1, 2\}$. □

Table 2: Functions of an ARBITER module.

y	$x = 00$	$x = 01$	$x = 10$	$x = 11$	$\lambda(y)$
0	$\{0\}$	$\{2\}$	$\{1\}$	$\{1,2\}$	00
1	$\{0\}$	$\{0\}$	$\{1\}$	$\{1\}$	10
2	$\{0\}$	$\{2\}$	$\{0\}$	$\{2\}$	01
	$\delta(x,y)$				

We can model any sequential machine of the Moore type using a module. In particular, delay elements (or wires with delays), logic gates, latches, counters, C-elements, and arbitration devices are included. Note that to hide the internal details of a module, we characterize it by a single multi-valued state variable. Also, we do not introduce any delays in the input and output wires, which allows us to model isochronic forks for the outputs. Outputs with anisochronic forks can be modeled within a network.

2.2 Network Behaviors

An asynchronous circuit is modeled by a "network" of modules. We consider only autonomous circuits, that is, circuits without external inputs. A circuit having external inputs can usually be transformed into an autonomous one by treating the environment of the circuit as a module.

For the purpose of connecting modules to form a network, we associate with each module a directed graph with m input vertices, all connected to one internal state vertex, which, in turn, is

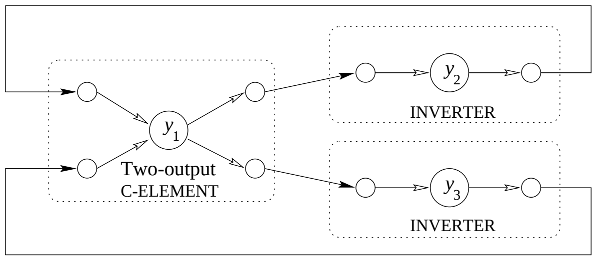

Figure 1: Network example: N_1.

connected to p output vertices. For example, see Figure 1. Informally, a *network* consists of a set of modules interconnected through *connections*; the set of connections form a *bijection* from the set of module outputs to the set of module inputs. These connections are delay-free in the sense that changes in the module outputs are copied to the corresponding module inputs instantaneously.

Example 2.3 Figure 1 shows a network N_1 composed of a two-output MULLER C-ELEMENT module and two INVERTER modules. A MULLER C-ELEMENT module synchronizes its two inputs; the two outputs follow the internal state without delay. An INVERTER module inverts its input. The connections of N_1 are marked by solid arrows. □

The state of a network N is entirely characterized by the states of its modules. Thus, a network state is represented by a tuple $s = (s_1, \ldots, s_n)$, where s_i is the state of the i-th module, $1 \leq i \leq n$. Starting in any network state, any unstable module may change its state according to its excitation. We assume that only one module changes state at a time. This is essentially the *general single-winner* (GSW) model [1]. The *GSW relation* over the set of states of N is a binary relation $\mathcal{R}$, such that $(s, t) \in \mathcal{R}$ if and only if t is reachable from s by changing the state of exactly one unstable module.

A *behavior* of the network N is an initialized graph $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$, where q is the initial state, $\mathcal{Q}$ is the set of states reachable from q via the GSW relation defined above, and $\mathcal{R}$ is the GSW relation restricted to $\mathcal{Q}$.

Example 2.4 Consider network N_1 shown in Figure 1. The behavior of N_1 with initial state $0\mathbf{00}$ is shown in Figure 2, where unstable state components are in boldface; also, from now on, when showing particular state tuples, we omit parentheses and commas, for brevity. Note the existence of *races* in the behavior; these are conditions in which more than one state variable is unstable. □

3 Network Delay-Insensitivity

To determine whether a network is delay-insensitive, we begin with its behavior in the presence of arbitrary component delays, and take this behavior to be correct. We wish to examine the effects of delays in the network connections. The motivation behind our approach is Molnar's "foam-rubber wrapper" postulate [9], where a module is viewed as being surrounded by a flexible boundary to reflect the fact that communication between its terminals and its environment takes place through a medium involving unknown delays.

Molnar's intuitive idea corresponds to surrounding each module in a network by DELAY mod-

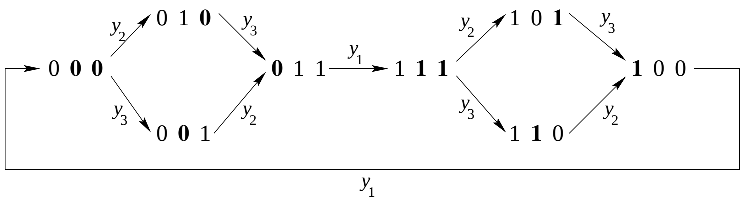

Figure 2: A behavior of network N_1.

ules. However, instead of "wrapping" the modules altogether, we adapt an inductive approach. A *delay extension* $\hat{N}$ of a network N is obtained by replacing some connections of N by DELAY modules. We refer to modules which are present in both N and $\hat{N}$ as the *original* modules. For the network N to be delay-insensitive, the behavior of *any* delay extension of N should satisfy a certain correctness concern with respect to the behavior of N. We define this correctness concern to correspond to *weak bisimulation* [8] (or *observation equivalence*), one of the strongest forms of behavioral equivalence. Transitions occurring on the inserted DELAY modules are treated as unobservable "internal" actions; in fact, they can be viewed simply as corresponding to the passing of time.

Let $\hat{N}$ be a delay extension of N. A state s of N and a state $\hat{s}$ of $\hat{N}$ are said to be *similar* if they agree on all the original modules. Let $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ be a behavior of N. A behavior $\hat{B} = \langle \hat{q}, \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ of $\hat{N}$ is said to be *initial-state compatible with* B if $\hat{q}$ is similar to q and all the inserted DELAY modules are stable in $\hat{q}$. In the following, $\hat{B}$ is always initial-state compatible with B, unless otherwise specified.

Let us call a change of a module state a *transition*. The behavior $\hat{B}$ is said to be *delay-safe with respect to* B if, for any similar states $\hat{s} \in \hat{\mathcal{Q}}$ and $s \in \mathcal{Q}$, whenever $\hat{t}$ is reachable from $\hat{s}$ by a sequence of changes involving exactly one transition on an original module, then s can reach t, a state similar to $\hat{t}$, by the same transition. On the other hand, if, whenever t is reachable from s by a certain transition, then $\hat{s}$ can reach $\hat{t}$, a state similar to t, by a sequence of changes involving that same transition along with changes on the inserted DELAY modules, then $\hat{B}$ is said to be *delay-faithful with respect to* B. Finally, $\hat{B}$ is said to be *livelock-free with respect to* B if no state of $\hat{B}$ can be reached from itself via a nonempty sequence of changes on the inserted DELAY modules.

A network N is *delay-insensitive with respect to a state* q if, for any delay extension $\hat{N}$ of N, $\hat{B}$ is delay-safe, delay-faithful, and livelock-free with respect to B. Delay-safety and delay-faithfulness constitute the bisimulative relationship between the behaviors B and $\hat{B}$. It can be shown that delay-faithfulness and livelock-freedom are satisfied by all delay extensions [2, 16]. Consequently, only delay-safety needs to be checked to verify delay-insensitivity.

Example 3.1 Consider part of a network N_2 shown in Figure 3(a) (for brevity, we hide the internal structures of the modules), where two DELAY modules are connected to an ARBITER module (as defined in Example 2.2). Figure 3(c) shows the corresponding part of a delay extension $\hat{N}_2$ of N_2, where the inserted DELAY module is shaded. Let the current values and excitations of the modules be as shown, with excitations shown in brackets.

Each figure represents a network state. State t of N_2, as shown in Figure 3(b), results from state s (Figure 3(a)) by a change on the unstable DELAY module. State $\hat{t}$ of $\hat{N}$ (Figure 3(d)) results from state $\hat{s}$ (Figure 3(c)) by the same change; also, $\hat{s}$ and s are similar. We assume that the states

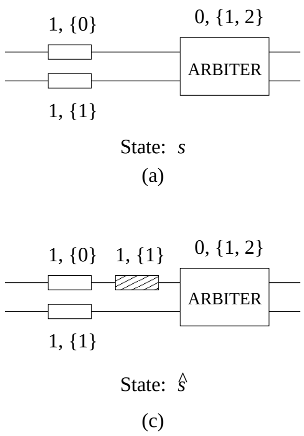

Figure 3: Example of delay-safety violation.

of network modules not shown do not change. Clearly, $\hat{t}$ and t are similar. In state $\hat{t}$, the ARBITER module may change to state 1 or 2, whereas in state t, it can only change to 2. This demonstrates a violation of delay-safety. Thus, N_2 is not delay-insensitive with respect to s. □

4 Semi-Modularity and Delay-Density

Semi-modularity of circuit behaviors was initially defined in [10] (see also [7]) for circuits in which each node has exactly one implied value (singleton excitation) at any time. The term "semi-modular" was adopted because a certain partially ordered set corresponding to such a behavior was shown to be a semi-modular lattice. In this paper, we use a more general definition [16] to suit our network model, where state variables may have several implied values at a time (e.g. in an ARBITER module), causing nondeterminism in network behaviors. Informally, semi-modularity requires that once a module M becomes unstable and b is in its excitation, b should remain in the excitation until M changes; in other words, an enabled action of a module cannot be disabled by the action of another module.

Example 4.1 For a violation of semi-modularity, refer to Figures 3(a) and (b). The change on the DELAY module involved alters the excitation of the ARBITER module and violates semi-modularity. □

Recall that wire delays are represented by DELAY modules. For the purpose of delay-insensitivity analysis, we must take into account both component delays and wire delays. Thus, to model an asynchronous circuit using a network, we associate a state variable with each component and connection wire. The resulting network is *delay-dense*, meaning that, of each pair of adjacent modules, at least one is a DELAY module. We take the behavior of this delay-dense network to be correct. The following theorems concerning delay-dense networks were proved in [2, 16].

Theorem 1 *For delay-dense networks, semi-modularity of behaviors is preserved under delay extensions.* □

Theorem 2 *A delay-dense network is delay-insensitive with respect to a state if and only if its behavior originating from that state is semi-modular.* □

5 The JTU-Rules

The first formal definition of delay-insensitivity was made by J. T. Udding in 1984 [13]. He used trace theory as the formalism for circuit specifications. An alphabet is a finite set of symbols. Finite-length sequences of symbols from an alphabet are called *traces*. A *trace structure* T is a triple $T = \langle \mathbf{i}T, \mathbf{o}T, \mathbf{t}T \rangle$, where $\mathbf{i}T$ is called the *input alphabet*, $\mathbf{o}T$ is the *output alphabet*, and $\mathbf{t}T$ is the *trace set*. A trace structure is viewed as the specification of a mechanism communicating with its environment. The input and output alphabets are two disjoint finite sets corresponding to the input and output terminals of the mechanism. An element a of $\mathbf{i}T$ (respectively, an element b of $\mathbf{o}T$) is interpreted as a signal at the input terminal a (respectively, output terminal b). The set $\mathbf{a}T = \mathbf{i}T \cup \mathbf{o}T$ is called the *alphabet* of T. A trace structure's trace set records all possible sequences of communication actions that can take place between the mechanism and its environment. Since each trace in a trace set represents a history of possible signals, any prefix of such a trace must itself be in the trace set. Thus, $\mathbf{t}T$ is *prefix-closed* in this sense, and always contains the empty word ϵ which represents the initial state in which no signal has yet been sent.

Following the foam-rubber wrapper postulate, Udding suggests that a delay-insensitive specification should satisfy the following two conditions. First, there should be no *computation interference*, meaning informally that no signal should be sent to a mechanism if that mechanism is not ready to receive it. Similarly, the mechanism should not send a signal to the environment, if the latter is not ready to receive it. Second, there should be no *transmission interference*, meaning informally that two consecutive signals cannot be sent along any wire. Udding then defines a trace structure $T = \langle \mathbf{i}T, \mathbf{o}T, \mathbf{t}T \rangle$ of a mechanism to be delay-insensitive if it satisfies the four rules (the JTU-Rules) stated below. Two symbols a and b in $\mathbf{a}T$ are said to be of the *same type* if they are both inputs or both outputs; otherwise, they are of *different types*. The rules are:

1. There should never be two consecutive signals along the same wire. Formally, for trace u and symbol $a \in \mathbf{a}T$, $uaa \notin \mathbf{t}T$.

2. There should not be any ordering between input signals of a mechanism, and the same holds for output signals. Formally, for traces u and v, and for symbols a and b of the same type, $uabv \in \mathbf{t}T$ if and only if $ubav \in \mathbf{t}T$.

3. If a and b are signals of different types, and both are enabled to occur after some trace u, then the occurrence of one signal should not disable the occurrence of the other signal. Formally, for trace u and symbols a and b of different types, if $ua \in \mathbf{t}T$ and $ub \in \mathbf{t}T$, then $uab \in \mathbf{t}T$.

4. Suppose signals a and b are of different types, and c is of the same type as a. If a and b can occur in either order at some point, and if c can occur when the order is ab, then it can also occur when the order is ba. Formally, for traces u and v, and for symbols a, b, and c with b of another type than a and c, if $uabvc \in \mathbf{t}T$ and $ubav \in \mathbf{t}T$, then $ubavc \in \mathbf{t}T$.

Udding has shown that, if a trace structure and its environment satisfy the four rules above, then there is no computation or transmission interference [13]. These four rules along with the prefix-closure property of the trace set constitute the class of delay-insensitive trace structures.

6 Main Theorem

We now study what semi-modularity of network behaviors imposes on the behaviors of the individual modules. Our main result states that all the (non-DELAY) modules in a delay-insensitive network must obey Udding's rules. We are concerned only with delay-dense networks, and we consider only networks containing more than one component. Thus, no module *feeds* itself in a network, where a module M is said to feed module M' if some output of M is connected to some input of M'. Furthermore, it is reasonable to assume that a transition on a module always brings the module to a stable state (we call this the *one-step* property of modules). The necessity of these assumptions will become clear in Section 7.

Consider a delay-dense network N and a module M which is not a DELAY module. Clearly, each input and output vertex of M is connected directly to a DELAY module. These DELAY modules can be seen as the input and output terminals of M, which form the interface between the module and its environment. Since DELAY modules carry only binary signals, the *observable behavior* of M is completely characterized by the set of possible sequences of transitions occurring on these terminals. We require that the network N be delay-dense just to make this characterization possible.

Definition 6.1 Let N be a delay-dense network, and let M be a module of N which is not a DELAY. Let the state variable corresponding to M be g. Define the following:

- The *input alphabet* of M: $\mathbf{i}M = \{a \in \mathcal{Y} \mid a \text{ feeds } g \text{ in } N\}$.
- The *output alphabet* of M: $\mathbf{o}M = \{b \in \mathcal{Y} \mid g \text{ feeds } b \text{ in } N\}$.
- The *alphabet* of M: $\mathbf{a}M = \mathbf{i}M \cup \mathbf{o}M$. $\square$

Clearly, all variables in $\mathbf{a}M$ correspond to DELAY modules. We make the assumption that $\mathbf{i}M \cap \mathbf{o}M = \emptyset$ for all M. Note that what we are assuming can be achieved through delay extensions. Since both delay-density and semi-modularity are preserved under delay extensions, we lose nothing by making such an assumption.

A behavior $B = \langle q, \mathcal{Q}, \mathcal{R}\rangle$ of a network N can be viewed as a nondeterministic finite automaton $\mathcal{B} = \langle \mathcal{Y}, \mathcal{Q}, q, \rho, \mathcal{Q}\rangle$, where $\mathcal{Y}$ is the input alphabet; $\mathcal{Q}$ is the state set; q is the initial state; ρ is the state transition function, $\rho : \mathcal{Q} \times \mathcal{Y} \to 2^{\mathcal{Q}}$, such that

$$\rho(s, a) = \{t \in \mathcal{Q} \mid t \text{ is reachable from } s \text{ via a change on } a\}$$

for $s \in \mathcal{Q}$ and $a \in \mathcal{Y}$; and $\mathcal{Q}$ is the set of accepting states. We denote the set of states reachable from a state s via a word $w \in \mathcal{Y}^*$ by the short-hand form sw. The language $L(\mathcal{B})$ accepted by $\mathcal{B}$ is defined as follows:

$$L(\mathcal{B}) = \{w \in \mathcal{Y}^* \mid qw \neq \emptyset\}.$$

It is worth noting that $L(\mathcal{B})$ is always prefix-closed. The language $L(\mathcal{B})$ records all possible (finite) sequences of state transitions allowed by the behavior. We alternately refer to this language as the *trace set* of the behavior B. Also, let us denote the projection of a string x (or a set of strings S) onto an alphabet α by $x \downarrow \alpha$ (or $S \downarrow \alpha$).

Definition 6.2 Let M and N be as defined in Definition 6.1. A *trace structure* of M is a triple $T = \langle \mathbf{i}M, \mathbf{o}M, \mathbf{t}M\rangle$ where $\mathbf{t}M \subseteq (\mathbf{i}M \cup \mathbf{o}M)^*$ is the *trace set*. Strings over the alphabet $\mathbf{a}M = \mathbf{i}M \cup \mathbf{o}M$ are called *traces*. Let B be a behavior of N. The *trace structure of M induced by B* is defined as $tr_M(B) = \langle \mathbf{i}M, \mathbf{o}M, L(\mathcal{B}) \downarrow \mathbf{a}M\rangle$. $\square$

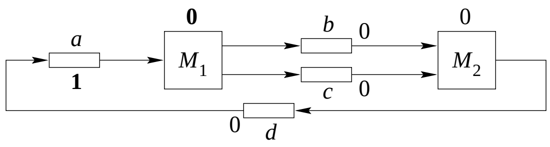

Figure 4: Network N_4.

Theorem 3 *Let M and N be as defined in Definition 6.1. Let $B = \langle q, \mathcal{Q}, \mathcal{R}\rangle$ be a semi-modular behavior of N, and let $tr_M(B) = \langle \mathbf{i}M,\ \mathbf{o}M,\ \mathbf{t}M\rangle$ be the trace structure of M induced by B. Then the trace set $\mathbf{t}M$is prefix-closed, and the JTU-Rules hold for M, i.e.,*

1. *For trace u and symbol $a \in \mathbf{a}M$, $uaa \notin \mathbf{t}M$.*
2. *For traces u and v, and for symbols $a, b \in \mathbf{a}M$ of the same type,* $uabv \in \mathbf{t}M \iff ubav \in \mathbf{t}M$.
3. *For trace u and symbols $a, b \in \mathbf{a}M$ of different types,* $(ua \in \mathbf{t}M \ \wedge\ ub \in \mathbf{t}M) \Longrightarrow uab \in \mathbf{t}M$.
4. *For traces u and v, and for symbol $b \in \mathbf{a}M$ of another type than $a, c \in \mathbf{a}M$,* $(uabvc \in \mathbf{t}M \ \wedge\ ubav \in \mathbf{t}M) \Longrightarrow (ubavc \in \mathbf{t}M)$. □

The behavior of an individual module operating within a network is captured by its trace structure induced by the behavior of the network. Theorem 3 basically states that if a behavior of a delay-dense network is semi-modular, which implies that the network is delay-insensitive by Theorem 2, then each "non-DELAY" module must behave delay-insensitively according to Udding's criterion. It is worth noting that the converse of this theorem is not true, as the following example demonstrates.

Example 6.1 Consider the network N_4 shown in Figure 4. Modules a, b, c, and d are DELAY modules. The functions of modules M_1 and M_2 are shown in Tables 3 and 4, respectively. We denote the internal state variables of M_1 and M_2 by g_1 and g_2, respectively.

Table 3: Functions of M_1.

g_1	a = 0	a = 1	bc
0	{2}	{1}	00
1	{1}	{1}	10
2	{2}	{2}	11
	$\delta(a, g_1)$		

Table 4: Functions of M_2.

g_2	bc = 00	01	10	11	d
0	{0}	{0}	{0}	{0}	0
	$\delta(bc, g_2)$				

Let the initial state be $ag_1bcg_2d = \mathbf{1}00000$. The corresponding behavior is shown in Figure 5. The trace structure of M_1 induced by the behavior is

$$\{\{a\}, \{b, c\}, \{\epsilon, a, b, ab, ba, ac, abc, acb\}\};$$

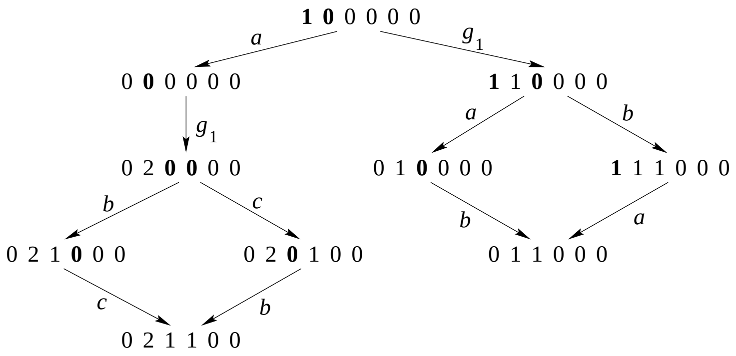

Figure 5: A behavior of network N_4.

and the trace structure of M_2 induced by the behavior is

$$\{\{b, c\}, \{d\}, \{\epsilon, b, c, bc, cb\}\}.$$

It is not hard to verify that both trace structures satisfy all the JTU-Rules. However, the network behavior is not semi-modular. Before the change of the DELAY module a, M_1 can change to state 1, but this is not possible after the change of a. By Theorem 2, N_4 is not a delay-insensitive network, even though all of its components are delay-insensitive in Udding's sense! □

7 Proof Sketch

In this section, we sketch the proof of the main theorem. A related result (the "Chain" Theorem) concerning the ordering among state transitions in a semi-modular behavior, is described first.

Let $B = \langle q, \mathcal{Q}, \mathcal{R}\rangle$ be a *semi-modular* behavior of a *delay-dense* network N. Let $\mathcal{B}$ be the nondeterministic finite automaton corresponding to B. Appearances of the same symbol in a string are ordered with the first appearance having index 1. In this section, superscripts are used exclusively to indicate appearance indices. For example, the string $u = abaaba$ implicitly represents $a^1b^1a^2a^3b^2a^4$. If a symbol a appears in a string y, we simply write $a \in y$; otherwise, we write $a \notin y$.

Definition 7.1 Let $u = xa^iyb^j \in L(\mathcal{B})$ for some $a, b \in \mathcal{Y}$ and $x, y \in \mathcal{Y}^*$. We say that a^i *leads to* b^j *in* u, denoted by $a^i \leadsto_u b^j$, if and only if

1. a feeds b in the network N, and
2. $a, b \notin y$. □

For example, consider the behavior in Figure 2. Let $u = y_3y_2y_1y_2y_3y_1$. Then we have $y_3^1 \leadsto_u y_1^1$, $y_2^1 \leadsto_u y_1^1$, and $y_1^1 \leadsto_u y_3^2$, among others. If u is implied by the context, we often omit u from $\leadsto_u$. The reflexive and transitive closure of $\leadsto$ is denoted by $\leadsto^*$.

Recall that by our assumption, no module feeds itself in a network, and all modules are one-step. Therefore, for any $u \in L(\mathcal{B})$, no two consecutive symbols in u are the same.

Proposition 7.1 *Let $r, t \in \mathcal{Q}$ and $a, b \in \mathcal{Y}$ be such that $t \in rab$. Then either a feeds b in N or $t \in rba$.*

Proof: Let us denote the value (excitation) of a variable c in a state s by s_c (S_c). Note that $a \neq b$. Let $r \xrightarrow{a} s \xrightarrow{b} t$ be a path in B, where $s \in \mathcal{Q}$. If a does not feed b in N, then a transition on a cannot change the excitation of b, i.e., $R_b = S_b$. Thus, we have $r \xrightarrow{b} s'$ for some $s' \in \mathcal{Q}$ such that $s'_b = t_b$. Note that a is unstable in state r. By semi-modularity, we have $R_a \subseteq S'_a$, and hence $s' \xrightarrow{a} t'$ for some $t' \in \mathcal{Q}$ such that $t'_a = s_a$. Clearly, t and t' are the same state; that is, $r \xrightarrow{b} s' \xrightarrow{a} t$, i.e., $t \in rba$. $\square$

Definition 7.2 Two strings $x, y \in \mathcal{Y}^*$ are said to be *P-similar* to each other, denoted by $x \sim y$, if some permutation of symbols in x forms y. $\square$

Theorem 4 ("Chain" Theorem) *Let $u = xa^iyb^jz \in L(\mathcal{B})$ for some $x, y, z \in \mathcal{Y}^*$ and $a, b \in \mathcal{Y}$. If $a, b \notin y$, then either $a^i \rightsquigarrow_u^* b^j$ or there exist $y', y'' \in \mathcal{Y}^*$ such that $xy'b^ja^iy''z \in L(\mathcal{B})$ and $y'y'' \sim y$.* $\square$

Corollary ("Cycle" Corollary) Let $u = xa^iya^jz \in L(\mathcal{B})$ for some $x, y, z \in \mathcal{Y}^*$ and $a \in \mathcal{Y}$. Then $a^i \rightsquigarrow_u^* a^j$. $\square$

The "Chain" Theorem can be seen as a generalization of Proposition 7.1. A proof of the theorem, which uses induction on the length of y, can be found in [16]. Note that the delay-density requirement is stronger than what we need for the theorem to hold. All we need is the non-existence of self-loops in the network.

Now, we present the proof of parts 1 and 2 of the main theorem. The rest of the proof can be found in [16].

Partial proof of Theorem 3: The trace set $\mathbf{t}M$ is prefix-closed because $L(\mathcal{B})$ is prefix-closed and prefix-closure is preserved under projections (see Property 1.2 of [13]).

1. Assume that $uaa \in \mathbf{t}M$. Then there exist $u', x \in \mathcal{Y}^*$ such that $w = u'a^ixa^{i+1} \in L(\mathcal{B})$ for some i, and $u' \downarrow \mathbf{a}M = u$ and $x \downarrow \mathbf{a}M = \epsilon$. By the "Cycle" Corollary, $a^i \rightsquigarrow_w^* a^{i+1}$. Note that $\mathbf{i}M \cap \mathbf{o}M = \emptyset$. By definition, there must be some $b \in \mathbf{a}M$, of type different from that of a, such that $b \in x$. This is a contradiction. Hence, $uaa \notin \mathbf{t}M$.

2. We only need to show that $uabv \in \mathbf{t}M \Longrightarrow ubav \in \mathbf{t}M$; the converse follows by symmetry. If $uabv \in \mathbf{t}M$, then there exist $u', x, v' \in \mathcal{Y}^*$ such that $w = u'a^ixb^jv' \in L(\mathcal{B})$, where $u' \downarrow \mathbf{a}M = u$, $v' \downarrow \mathbf{a}M = v$, and $x \downarrow \mathbf{a}M = \epsilon$. Clearly, $a, b \notin x$. By the "Chain" Theorem, we have two cases:

   - There exist $x', x'' \in \mathcal{Y}^*$ such that $w' = u'x'b^ja^ix''v' \in L(\mathcal{B})$ and $x'x'' \sim x$:
     Then $ubav = u'x'bax''v' \downarrow \mathbf{a}M = w' \downarrow \mathbf{a}M \in \mathbf{t}M$, as required.
   - $a^i \rightsquigarrow_w^* b^j$:
     Note that a and b are of the same type and $\mathbf{i}M \cap \mathbf{o}M = \emptyset$. By definition, there must be some $c \in \mathbf{a}M$, of type different from that of a and b, such that $c \in x$, which is a contradiction.

   Hence, $uabv \in \mathbf{t}M \Longleftrightarrow ubav \in \mathbf{t}M$. $\square$

8 Conclusion

In this paper, we have shown that for delay-dense networks, semi-modularity implies that each module in the network communicates delay-insensitively with its environment. The fact that the converse of this result is not true (as shown in Example 6.1) and that semi-modularity and network delay-insensitivity are equivalent for delay-sense networks demonstrates that there are situations where each component of a network behaves delay-insensitively, while the network is not delay-insensitive. This leads us to believe that the JTU rules need to be strengthened to resolve this oddity.

References

[1] J. A. Brzozowski and C-J. Seger, *Asynchronous Circuits,* Springer-Verlag, New York, NY, 1995.

[2] J. A. Brzozowski and H. Zhang, "Delay-Insensitivity and Semi-Modularity," Technical Report CS-97-11, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, March 1997. Available via anonymous ftp at: ftp://maveric.uwaterloo.ca/pub/reports/John.Brzozowski/DISM.ps.gz.

[3] T. J. Chaney and C. E. Molnar, "Anomalous Behavior of Synchronizers and Arbiters," *IEEE Trans. on Computers*, C-22: pp. 421–422, 1973.

[4] D. L. Dill, *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*, The MIT Press, Cambridge, Massachusetts, 1988.

[5] J. C. Ebergen, *Translating Programs into Delay-Insensitive Circuits,* Ph.D. Thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands, October 1987. Also, CWI Tract 56, Center for Mathematics and Computer Science, Amsterdam, The Netherlands, 1989.

[6] C. Mead and L. Conway, *Introduction to VLSI Systems,* Addison-Wesley, Reading, MA, 1980.

[7] R. E. Miller, "Speed-Independent Switching Circuit Theory," Chapter 10 in *Switching Theory, Volume II: Sequential Circuits and Machines,* John Wiley & Sons, 1965.

[8] R. Milner, *Communication and Concurrency,* Prentice Hall, 1989.

[9] C. E. Molnar, T. P. Fang, and F. U. Rosenberger, "Synthesis of Delay-Insensitive Modules," *Proceedings of the 1985 Chapel Hill Conference on VLSI,* H. Fuchs, ed., Computer Science Press, Rockville, Maryland, pp. 67–86, 1985.

[10] D. E. Muller and W. C. Bartky, "A Theory of Asynchronous Circuits," in *Annals of Computing Laboratory of Harvard University*, pp. 204–243, 1959.

[11] C. L. Seitz, "System Timing," Chapter 7 in [6], pp. 218–262.

[12] N. Shintel and M. Yoeli, "Synthesis of Modular Networks from Petri-Net Specifications," Technical Report #743, Israel Institute of Technology, Haifa 32000 Israel, 1992.

[13] J. T. Udding, *Classification and Composition of Delay-Insensitive Circuits,* Doctoral Dissertation, Eindhoven University of Technology, Eindhoven, The Netherlands, 1984.

[14] J. T. Udding, "A Formal Model for Defining and Classifying Delay-Insensitive Circuits and Systems," *Distributed Computing,* vol. 1, no. 4, pp. 197–204, 1986.

[15] T. Verhoeff, "Characterizations of Delay-Insensitive Communication Protocols," Computing Science Notes No. 89/06, Department of Mathematics and Computing Science, Eindhoven University of Technology, May 1989.

[16] H. Zhang, *Delay-Insensitive Networks,* Master's Thesis, Department Computer Science, University of Waterloo, Waterloo, Ontario, Canada, June 1997. Available via anonymous ftp at: ftp://maveric.uwaterloo.ca/pub/reports/All-reports/HZ_thesis.ps.Z.