DELAY-INSENSITIVE NETWORKS

by

Hao Zhang

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 1997

©Hao Zhang 1997

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

An asynchronous network 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. In this thesis, we study delay-insensitivity of autonomous networks, i.e., networks without external inputs, whose components can be arbitrary sequential machines of the Moore type. A formal model for this type of network and their behavior is established. We introduce a new definition of delay-insensitivity which uses the notion of behavioral equivalence in the presence of arbitrary delays. In particular, 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 to the behavior of N. We relate the delay-insensitivity of a network to the concept of semi-modularity of its behavior. Roughly speaking, a behavior is semi-modular if an enabled action cannot be disabled by another action. We generalize the existing definition of semi-modularity (due to Muller) to reflect nondeterminism in network behaviors. We prove that 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. Furthermore, we prove several results relating delay-insensitivity to semi-modularity, among which is the equivalence between delay-insensitivity and semi-modularity for delay-dense networks, where a network is delay-dense if, of each pair of adjacent components in the network, at least one is a delay element. A discussion of the problem of testing whether a given network is delay-insensitive is also included.

Acknowledgements

First of all, I wish to thank Prof. John Brzozowski, my supervisor, who not only introduced me to this interesting topic, but also made my research in the area such a fun and rewarding learning experience. In regards to my thesis, he has helped me greatly in establishing the network model and finding new problems to solve. I owe him a great deal for his constructive suggestions and ideas. And his words of encouragement have been the constant driving force behind my work. Last but not the least, I would like to thank him for the numerous times that he has proofread my writing. My thanks also go to the three readers of my thesis: Prof. Larry Cummings, Prof. Jo Ebergen, and Prof. Jeff Shallit. I appreciate the time and effort they have spent and the valuable comments they have given to me. I wish to thank all the members of the MAVERIC research group for their helpful criticisms and suggestions at several presentations of my research. Finally, I gratefully acknowledge the financial support provided by the National Science and Engineering Research Council of Canada, under grant No. OGP 0000871, and by the Information Technology Research Centre of Ontario.

献给我父母：张润楚和肖云茹

Contents

1 Introduction **1**

2 Network Model **7**

2.1 Modules 7

2.2 Networks 14

2.3 Network Behaviors 17

3 Delay-Insensitivity **22**

3.1 Delay Extensions 22

3.2 Delay-Insensitivity of Networks 25

4 Semi-Modularity **30**

4.1 Generalization of Semi-Modularity 30

4.2 Semi-Modularity and Delay-Density 33

4.2.1 Delay-Dense Networks 33

4.2.2 Semi-Modular Delay-Dense Networks 34

4.3 Semi-Modularity and the JTU-Rules 39

4.3.1 The "Chain" Theorem 40

4.3.2 The JTU-Rules 43

4.3.3 Semi-Modularity Implies the JTU-Rules 45

5 Delay-Insensitivity and Semi-Modularity **56**

5.1 Delay-Insensitivity and Semi-Modularity 56

5.2 Testing Delay-Insensitivity of Networks 60

6 Concluding Remarks **63**

6.1 Open Problems 63

6.2 Classification of Asynchronous Networks 64

References **68**

List of Tables

2.1 Functions of a DELAY module. 10

2.2 Functions of an INVERTER module. 11

2.3 Functions of a FORK module. 11

2.4 Functions of a three-input MAJORITY module. 11

2.5 Functions of an SR-LATCH module. 12

2.6 Functions of a MOD-4 COUNTER module. 12

2.7 Functions of a C-ELEMENT module. 13

2.8 Functions of an ARBITER module. 13

2.9 Functions of a module which is not one-step. 14

4.1 Functions of M_1. 54

4.2 Functions of M_2. 54

List of Figures

2.1 Module graph of a DELAY module. 10

2.2 Network example: N_1. 16

2.3 Network example: N_2. 16

2.4 A behavior of network N_1. 19

2.5 A behavior of network N_2. 19

3.1 Example of delay-safety violation. 28

4.1 Network example: N_3. 32

4.2 A semi-modular behavior of network N_3. 32

4.3 Theorem 4.2 fails to hold for networks which are not delay-dense. . 39

4.4 Pictorial proof of Proposition 4.3.2 and its corollary. 41

4.5 Network N_4. 54

4.6 A behavior of network N_4. 55

5.1 Theorem 5.3 fails to hold for networks which are not delay-dense. . 59

6.1 Classification of asynchronous networks. 64

6.2 Network examples for Figure 6.1. 65

6.3 Classification of deterministic asynchronous networks. 66

6.4 Network examples for Figure 6.3. 67

Chapter 1

Introduction

Although much of today's digital design is based on a synchronous approach, there has been a considerable interest in asynchronous circuits, especially during the past decade. 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 [17]).

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, 5, 6, 7, 13, 15, 18, 19, 20, 22, 23]. One of the advantages of such circuits is that they allow certain flexibility in circuit design. For example, in a delay-insensitive circuit, it is possible to replace some components by faster or slower ones, or change the material used for connection wires, without changing the correctness of the circuit operation.

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 [17]. This scaling problem has become increasingly troublesome in VLSI design, as it is often desirable to map the existing layout of a circuit onto a smaller region as technology improves. Moreover, the metastability phenomenon [4] is a manageable problem in delay-insensitive circuits. A circuit is in a metastable state if it is expected to choose between mutually-exclusive states, but the decision may be delayed indefinitely; such a delay will not affect the correct operation of a delay-insensitive circuit.

We now briefly describe some previous work on delay-insensitivity. Readers may wish to refer to [1] for a more complete survey.

Early work on circuit behaviors in the presence of arbitrary delays traces back to D. E. Muller's theory of "speed-independent" circuits [11, 15]. He considered *autonomous* circuits, i.e., circuits without external inputs. He assumed that there are delays only in the circuit components, but there are no delays in the interconnection lines of the circuit. The behavior of a circuit is described by a set of *allowed sequences* of states, which specify the order in which the state variables change, but not the times of change. Muller's definition of speed-independence is related to the *terminal classes* (or outcome [3]) of the allowed sequences for a circuit. We will come back to this in Section 6.2.

S. H. Unger studied how the delays in logic elements and connection wires (*stray delays*) may affect the behaviors of asynchronous sequential circuits [21]. He assumed that stray delays may have values ranging from 0 to some known upper bound, and the so-called *delay elements* can have some minimum value specified by the designer. Unger dealt with certain limitations of delay-insensitive circuits.

He showed that no asynchronous behavior with an *essential hazard* can be realized by a circuit composed of logic gates without added delay elements. An essential hazard is present when there is a state s and a binary input a such that, if the circuit is initially in s and a is changed once, the state reached is *not* the same as the one reached if a changed two more times. With added delay elements, however, the realization is possible.

Relating to limitations of delay-insensitivity, J. A. Brzozowski and J. C. Ebergen showed that such commonly used behaviors as those of the set-reset latch and Muller's C-element do not have delay-insensitive realizations, if logic gates are used as the basic components and the circuits are operating in the so-called *input-output mode*, where inputs may change in response to output changes even though the circuit has not been stabilized [2]. Their proof makes use of ternary simulation, which can also be used to prove Unger's result mentioned above [16].

C. E. Molnar et al. introduced the "foam-rubber wrapper" postulate [13] 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 of delay-insensitivity [19, 20]; this definition is based on trace theory. He suggested that a specification 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 4.3.2.

J. C. Ebergen described a formal approach to the design of delay-insensitive circuits using *decompositions* of asynchronous circuit components into sets containing only certain basic components [6, 7], such as fork, Muller's C-element, and toggle. Each component of an asynchronous circuit is specified by a regular expression-like program which characterizes the communication behavior between the component and its environment. Such a program can be translated into a set of programs, each representing the behavior of some basic component. The network of these basic components forms a realization, called the decomposition, of the original component. If all the basic components used are so-called delay-insensitive components, then the decomposition is said to be delay-insensitive and the correctness of the network is insensitive to delays in components and connection wires. Ebergen has proved that his definition of delay-insensitivity is equivalent to that of Udding's.

D. L. Dill developed methods for the specification and automatic verification of asynchronous circuits [5]. He used a more general notion of trace structures, called the *prefix-closed trace structures*, which consists of a set of *successful traces* and a set of *failure traces*, in addition to input and output alphabets. Trace structures are required to be *receptive*, meaning that the component is ready to accept input from the environment at any time. A trace structure S is said to *conform to* a trace structure T if S can *safely* substitute for T in a circuit; safe substitution preserves the *failure-freedom* of the trace structures, where a trace structure is failure-free if its failure set is empty. Dill defined delay-insensitivity via an operator **DI** on trace structures. Basically, **DI** attaches delays to all the inputs and outputs of a trace structure (a foam-rubber wrapper). A trace structure T is said to be delay-insensitive if and only if $\mathbf{DI}(T)$ conforms to T. In Dill's model, the problem of determining whether a trace structure is delay-insensitive is decidable.

T. Verhoeff called an asynchronous circuit component a *process* [22, 23]. A process is specified by a trace structure, as used by Udding. Verhoeff has shown that several characterizations of delay-insensitivity are equivalent under certain assumptions, among which are the JTU-Rules and the so-called "neighbor-swap rule" [22]. He also extended the JTU-Rules to include progress as a correctness concern.

From what we have seen so far, definitions of delay-insensitivity have dealt mainly with specifications of asynchronous circuits using trace structures or regular expression-like programs, 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 thesis, we consider delay-insensitivity of autonomous networks whose components can be any sequential machines of the Moore type [14]. A formal model for this type of networks and their behaviors is established. We define delay-insensitivity of networks 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 [12] to the behavior of N, where transitions occurring on the added delays are considered unobservable. N. Shintel and M. Yoeli [18] have used a similar delay-addition scheme in their definition of delay-insensitive realization. Roughly speaking, they define a network N to be a delay-insensitive implementation of a behavior B if the trace set of the network $\hat{N}$, obtained from N by surrounding each component of N by delays, projects onto the trace set of B, and also N does not have deadlock.

Semi-modularity of circuit behaviors was initially studied by Muller [11, 15]. In Muller's terminology, a node i is *excited* in state s if its current value s_i disagrees with its *implied value* (we call it the *excitation* of the node) s_i'; the implied value is

determined by the Boolean expressions associated with the node. Muller considered deterministic circuit behaviors, where a behavior is deterministic if each node has exactly one implied value at any time. In a semi-modular circuit, if a node i is excited in state s, but node i does not change to s_i' when the circuit goes to state t, then node i must still be excited in state t and to the same value as in state s. So roughly speaking, semi-modularity requires that no state transition can change the implied value of any unchanged state variable which is unstable. More recently, A. J. Martin [8] used the term "stability", instead of semi-modularity, and basically adopted it as a definition of delay-insensitivity [9].

In this thesis, we generalize Muller's definition of semi-modularity to reflect non-determinism in network behaviors. We prove that for a special class of networks, the delay-dense networks, semi-modularity is preserved through the addition of delays, where a network is delay-dense if, of each pair of adjacent network components, at least one is a delay element. We relate semi-modularity to the JTU-Rules by proving that 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 the JTU-Rules. Furthermore, we prove several results relating delay-insensitivity to semi-modularity, among which is the equivalence between delay-insensitivity and semi-modularity for delay-dense networks. A discussion of the problem of testing whether a given network is delay-insensitive is also included.

The thesis is structured as follows. In Chapter 2, we present our model for asynchronous components, networks, and network behaviors. Chapter 3 concentrates on our definition of network delay-insensitivity. In Chapter 4, we generalize the existing definition of semi-modularity, and prove new results concerning semi-modularity. Chapter 5 includes proofs of several results relating delay-insensitivity to semi-modularity. Chapter 6 concludes the thesis.

Chapter 2

Network Model

2.1 Modules

We first introduce our model of asynchronous component, which we call a "module." To hide the details of the internal design of a module, we characterize it only by an abstract internal state. This also helps to keep the model simple. We do not introduce any delays in the input and output wires of the module for several reasons. First, wire delays are frequently ignored in modules designed on a very small area of a chip; these areas, called the *equipotential regions*, are sufficiently small that the delay associated with equalizing the potential across any wire is small in comparison with switching delays or propagation delays between modules [17]. Second, we want to have the ability to model *isochronic* forks and similar components, since they are used in many practical designs. A fork is isochronic if all the output branches of the fork change at exactly the same time. And finally, if we do wish to have modules with input and output delays, we can model them within a network, which is defined in the next section.

Definition 2.1.1 A *module* is a pair $M^i = \langle C^i, G^i \rangle$, where

- $C^i = \langle \mathcal{S}^i, \mathcal{X}^i, y^i, \mathcal{Z}^i, \delta^i, \lambda^i \rangle$ is a *sequential machine*, where
  - $\mathcal{S}^i$ is a finite set of *module internal states*;
  - $\mathcal{X}^i = \{x^i_1, \ldots, x^i_{m^i}\}$, $m^i \geq 1$, is the set of *module input variables*; also, $x^i = (x^i_1, \ldots, x^i_{m^i})$ is the m^i-tuple of module input variables;
  - y^i is the *module internal state variable*;
  - $\mathcal{Z}^i = \{z^i_1, \ldots, z^i_{p^i}\}$, $p^i \geq 1$, is the set of *module output variables*; also, $z^i = (z^i_1, \ldots, z^i_{p^i})$ is the p^i-tuple of module output variables;
  - δ^i is the *module excitation function*, $\delta^i : \{0,1\}^{m^i} \times \mathcal{S}^i \rightarrow (2^{\mathcal{S}^i} \backslash \emptyset)$, and for any $a \in \{0,1\}^{m^i}$ and $b \in \mathcal{S}^i$, either $\delta^i(a,b) = \{b\}$ or $b \notin \delta^i(a,b)$;
  - $\lambda^i = (\lambda^i_1, \ldots, \lambda^i_{p^i})$ is the *module output function*, $\lambda^i : \mathcal{S}^i \rightarrow \{0,1\}^{p^i}$;
- $G^i = \langle \mathcal{V}^i, \mathcal{E}^i \rangle$ is a directed graph, the *module graph*, where
  - $\mathcal{V}^i$ is the set of *module vertices*, partitioned into $\mathcal{I}^i$, $\mathcal{J}^i$, and $\mathcal{O}^i$, where
    - $\mathcal{I}^i$ is the set of *module input vertices*, which are labelled $x^i_1, \ldots, x^i_{m^i}$, corresponding to the module input variables;
    - $\mathcal{J}^i$ is the singleton set of *module internal state vertex*, which is labelled y^i, corresponding to the module internal state variable;
    - $\mathcal{O}^i$ is the set of *module output vertices*, which are labelled $z^i_1, \ldots, z^i_{p^i}$, corresponding to the module output variables;
  - $\mathcal{E}^i = (\mathcal{I}^i \times \mathcal{J}^i) \cup (\mathcal{J}^i \times \mathcal{O}^i)$ is the set of *module edges*. □

An *input state* of the module is a tuple $a \in \{0,1\}^{m^i}$, an *internal state* of the module is an element $b \in \mathcal{S}^i$, and a *total state* is an ordered pair $(a,b) \in \{0,1\}^{m^i} \times \mathcal{S}^i$. Let $t = (a,b)$ be a total state. The set $T = \delta^i(t) \subseteq \mathcal{S}^i$ is the *excitation state set*, or simply, the *excitation* of the module in state 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. If the cardinality of T is always 1, the module is said to be *deterministic*. In that case, we view δ^i as a function from the set of total states to $\mathcal{S}^i$. The *output state* of the module, which is a binary p^i-tuple, is determined solely by the present internal state. If the internal state changes, and the output state computed by the output function differs from the output state before the change, the new output value appears *instantaneously* together with the internal state change.

Note that the sets $\mathcal{I}^i$, $\mathcal{J}^i$, and $\mathcal{O}^i$ are disjoint. Thus, the module graph does not have self-loops. One may view the module graph as a dependency graph for the variables associated with a module, with the self-dependency arc on the internal state variable missing.

Definition 2.1.1 permits us to treat any sequential machine of the Moore type [14] as a module. In particular, delay elements (or wires with delays), forks, logic gates, latches, counters, C-elements, and arbitration devices (they play important roles in most concurrent systems) are included. Note that in this model, forks are isochronic. *Anisochronic* forks can be modelled within a network.

Now let us look at some examples of modules. If only one module is being discussed, we often drop the superscript i, for convenience. The modules in the first six examples are deterministic.

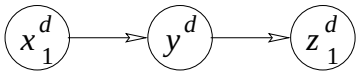


Figure 2.1: Module graph of a DELAY module.

Table 2.1: Functions of a DELAY module.

$\delta^d(x, y)$

	x		
y	0	1	$\lambda^d(y)$
0	0	1	0
1	0	1	1

Example 2.1.1 A DELAY module is defined as $M^d = \langle C^d, G^d \rangle$, with $C^d = \langle \{0,1\}, \{x_1^d\}, y^d, \{z_1^d\}, \delta^d, \lambda^d \rangle$, and δ^d and λ^d defined as in Table 2.1. Its module graph is shown in Figure 2.1. In the sequel, we often use the superscript d to identify a DELAY module.

Example 2.1.2 An INVERTER module has $\mathcal{S} = \{0,1\}$, $m = 1$, $p = 1$, and δ and λ defined as in Table 2.2.

Example 2.1.3 An (isochronic) FORK module has $\mathcal{S} = \{0,1\}$, $m = 1$, $p = 2$, and δ and λ defined as in Table 2.3.

Example 2.1.4 A three-input MAJORITY module has $\mathcal{S} = \{0,1\}$, $m = 3$, $p = 1$, and δ and λ defined as in Table 2.4.

Example 2.1.5 A (set-dominant) SR-LATCH module has $\mathcal{S} = \{0,1\}$, $m = p = 2$, and δ and λ defined as in Table 2.5.

Table 2.2: Functions of an INVERTER module.

	x		
y	0	1	$\lambda(y)$
0	1	0	0
1	1	0	1
	$\delta(x,y)$		

Table 2.3: Functions of a FORK module.

	x		
y	0	1	$\lambda(y)$
0	0	1	00
1	0	1	11
	$\delta(x,y)$		

Table 2.4: Functions of a three-input MAJORITY module.

	x								
y	000	001	010	011	100	101	110	111	$\lambda(y)$
0	0	0	0	1	0	1	1	1	0
1	0	0	0	1	0	1	1	1	1
	$\delta(x,y)$								

Table 2.5: Functions of an SR-LATCH module.

	x				
y	00	01	10	11	$\lambda(y)$
0	0	0	1	1	01
1	1	0	1	1	10
	$\delta(x,y)$				

Table 2.6: Functions of a MOD-4 COUNTER module.

	x		
y	0	1	$\lambda(y)$
0	0	1	00
1	2	1	01
2	2	3	10
3	0	3	11
	$\delta(x,y)$		

Example 2.1.6 A MOD-4 COUNTER module has $\mathcal{S} = \{0,1,2,3\}$, $m = 1$, $p = 2$, and δ and λ defined as in Table 2.6.

Example 2.1.7 A (Muller) C-ELEMENT module has $\mathcal{S} = \{0,1\}$, $m = 2$, $p = 1$, and δ and λ defined as in Table 2.7.

Example 2.1.8 An ARBITER module has $\mathcal{S} = \{0,1,2\}$, $m = p = 2$, and δ and λ defined as in Table 2.8. ARBITER modules are 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 2.7: Functions of a C-ELEMENT module.

	x				
y	00	01	10	11	$\lambda(y)$
0	0	0	0	1	0
1	0	1	1	1	1
	$\delta(x,y)$				

Table 2.8: Functions of an ARBITER module.

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

Table 2.9: Functions of a module which is not one-step.

	x		
y	0	1	$\lambda(y)$
0	$\{2\}$	$\{1,2\}$	0
1	$\{0\}$	$\{2\}$	0
2	$\{2\}$	$\{1\}$	1
	$\delta(x,y)$		

Definition 2.1.2 A module M^i, as in Definition 2.1.1, is said to be *one-step* if, for all $a \in \{0,1\}^{m^i}$ and $b \in \mathcal{S}^i$, whenever $b' \in \delta^i(a,b)$, then $\delta^i(a,b') = \{b'\}$.

All the modules we have seen so far are one-step. Modules which are not one-step are not often used in practice. However, at this point, there is no reason to restrict our model to one-step modules (The only section in which we make the assumption that all modules are one-step is Section 4.3). Table 2.9 shows the functions of a module $M^{\updownarrow}$ (the superscript is used to signify the module's possible oscillatory behavior) which is not one-step. When the input is fixed at 1, the state of $M^{\updownarrow}$ will eventually oscillate between 1 and 2.

2.2 Networks

We now introduce our model of asynchronous circuit, which we call a "network." 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.

Definition 2.2.1 A *network* is a pair $N = \langle \mathcal{M}, G \rangle$, where

- $\mathcal{M} = \{M^1, \ldots, M^n\}$, $n \geq 1$, is a set of modules, where M^i is as defined in Definition 2.1.1 for $1 \leq i \leq n$;
- $G = \langle \mathcal{V}, \mathcal{E} \rangle$ is a connected directed graph, the *network graph*, where
  - $\mathcal{V} = \bigcup_{i=1}^{n} \mathcal{V}^i$ is the set of *network vertices*;
  - $\mathcal{E} \supseteq \bigcup_{i=1}^{n} \mathcal{E}^i$ is the set of *network edges*, such that

$$\mathcal{E} \subseteq (\bigcup_{i=1}^{n} \mathcal{O}^i \times \bigcup_{i=1}^{n} \mathcal{I}^i) \cup \bigcup_{i=1}^{n} \mathcal{E}^i,$$

    and

    1. for each module input vertex $x_l^j \in \mathcal{V}$, there exists exactly one module output vertex $z_k^i \in \mathcal{V}$ such that $(z_k^i, x_l^j) \in \mathcal{E}$;
    2. for each module output vertex $z_k^i \in \mathcal{V}$, there exists exactly one module input vertex $x_l^j \in \mathcal{V}$ such that $(z_k^i, x_l^j) \in \mathcal{E}$. □

We denote the set of network edges that are not internal module edges by $\mathcal{K}$, and refer to these edges as *connections*. The set of connections form a *bijection* from the set of module output vertices to the set of module input vertices.

Example 2.2.1 Figure 2.2 shows a network N_1 composed of a C-ELEMENT module, an INVERTER module, and a FORK module, where connections are marked by solid arrows and module edges are marked by hollow arrows.

Example 2.2.2 Figure 2.3 shows a somewhat similar network N_2 composed of a two-output C-ELEMENT module and two INVERTER modules.

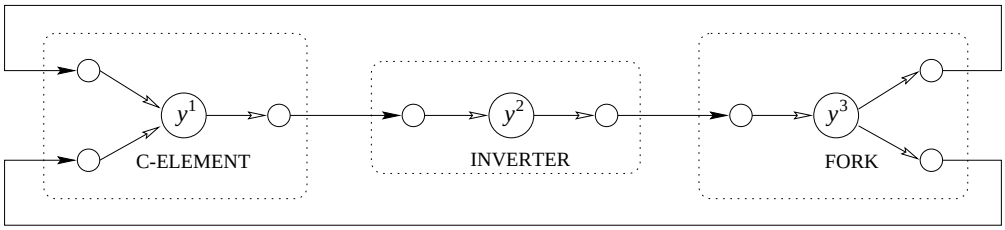


Figure 2.2: Network example: N_1.

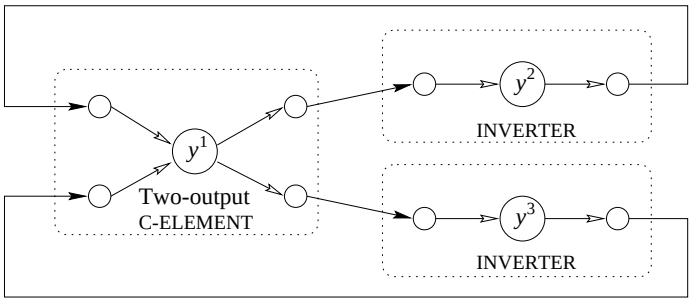


Figure 2.3: Network example: N_2.

2.3 Network Behaviors

Given a network N, as in Definition 2.2.1, the set of *state variables* of N is $\mathcal{Y} = \{y^1, \ldots, y^n\}$. The set of *states* of N is $\mathcal{S} = \mathcal{S}^1 \times \ldots \times \mathcal{S}^n$. If $s \in \mathcal{S}$, the i-th component of s is denoted by s_i. In general, we adopt the convention that specific components of a tuple are indexed using subscripts.

Definition 2.3.1 Let $y^i \in \mathcal{Y}$. The *network excitation function* $\Delta_i : \mathcal{S} \to 2^{\mathcal{S}^i}$ of y^i, is defined as follows,

$$\Delta_i(s) = \delta^i(\lambda^{h_1}_{k_1}(s_{h_1}), \ldots, \lambda^{h_{m^i}}_{k_{m^i}}(s_{h_{m^i}}), s_i)$$

where $(z^{h_j}_{k_j}, x^i_j)$ is a connection for $1 \leq j \leq m^i$.

In fact, Δ_i is the module excitation function δ^i of M^i with arguments changed as follows: If the k-th output of module M^h is connected to the j-th input of M^i, then the j-th input argument of δ^i becomes the function $\lambda^h_k(y^h)$. Since λ^h depends solely on y^h, the excitation function Δ_i becomes a function of y^h.

Definition 2.3.2 For $y^i \in \mathcal{Y}$ and $s \in \mathcal{S}$, the *excitation* of y^i in state s, denoted by S_i, is $S_i = \Delta_i(s)$.

Definition 2.3.3 A state s is *stable* if $S_i = \{s_i\}$ for all i; otherwise, s is *unstable*. We denote the set of unstable state variables in state s by

$$\mathcal{U}(s) = \{y^i \in \mathcal{Y} \mid S_i \neq \{s_i\}\}.$$

We now define the behavior of a network. A major assumption we make here is that only one state variable changes at a time. This is essentially the *general single-winner* (GSW) model [3]. One motivation behind this assumption, paraphrasing

from [2], follows from the fact that we are mainly interested in delay-insensitive networks. For such networks, wire delays have to be considered, which makes it difficult to argue strongly in favor of a model where simultaneous changes play a significant role. Semi-modularity is the other network property we are mostly concerned with. Muller [15] showed that both the *general multiple-winner* model, in which simultaneous variable changes are allowed, and the general single-winner model produce the same result for a network with a semi-modular behavior. This is because semi-modularity requires that no unstable state variable can become stable without changing its value. Thus, variables changing simultaneously can always be made to change sequentially. Of course, simplicity of the general single-winner model is also a factor.

Definition 2.3.4 The *GSW relation* over the set $\mathcal{S}$ of states of a network N is a binary relation $\mathcal{R}$, such that $(s,t) \in \mathcal{R}$, or $s\mathcal{R}t$, if and only if s differs from t in exactly one component i (that is, $s_i \neq t_i$), $y^i \in \mathcal{U}(s)$, and $t_i \in S_i$. The reflexive and transitive closure of $\mathcal{R}$ is denoted by $\mathcal{R}^*$.

Definition 2.3.5 A (GSW) *behavior* of a network N is an initialized directed graph $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$, where

- $q \in \mathcal{S}$ is the initial state;
- $\mathcal{Q}$, specifying the vertices of B, is the set of states reachable from q via the GSW relation $\mathcal{R}$,
$$\mathcal{Q} = \{s \in \mathcal{S} \mid q\mathcal{R}^* s\};$$
- $\mathcal{R}$, specifying the edges of B, is the GSW relation of Definition 2.3.4 restricted to $\mathcal{Q}$.

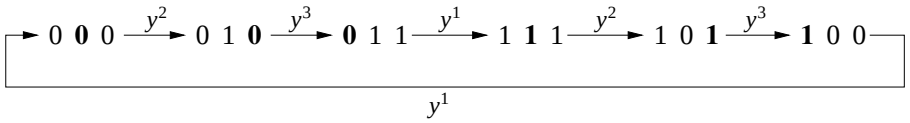


Figure 2.4: A behavior of network N_1.

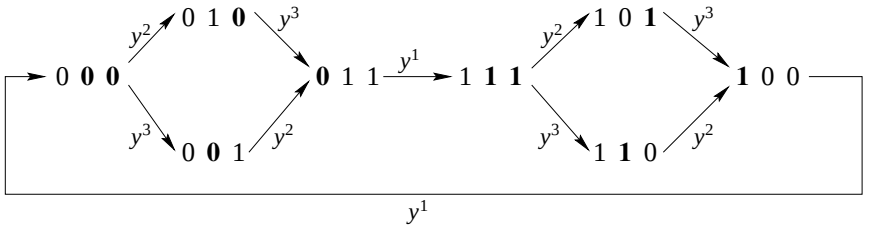


Figure 2.5: A behavior of network N_2.

If $(s,t) \in \mathcal{R}$, we attach to edge (s,t) a *tag* $\tau(s,t) \in \mathcal{Y}$, which denotes the state variable in which s and t differ. □

Example 2.3.1 Consider network N_1 shown in Figure 2.2. The set of state variables is $\mathcal{Y} = \{y^1, y^2, y^3\}$. The behavior of N_1 with initial state $(0,0,0)$ is shown in Figure 2.4, where unstable state components are in boldface; also, from now on, when showing particular state tuples, we omit parentheses and commas, for brevity. We see that this behavior is *sequential*.

Example 2.3.2 Consider network N_2 shown in Figure 2.3. The behavior of N_2 with initial state 000 is shown in Figure 2.5. Note the existence of *races* in the behavior; these are conditions in which more than one state variable is unstable.

Definition 2.3.1 reflects a functional dependency among the modules of a network. This is captured by the "feed" relation, defined as follows:

Definition 2.3.6 A module M^i is said to *feed* a module M^j in the network N, denoted by $(M^i, M^j) \in \mathcal{F}$, if there exist $z^i_k \in \mathcal{O}^i$ and $x^j_l \in \mathcal{I}^j$ such that $(z^i_k, x^j_l) \in \mathcal{K}$.

The relation $\mathcal{F}$ can also be viewed as being defined over the set of state variables. That is, $(y^i, y^j) \in \mathcal{F}$ if and only if $(M^i, M^j) \in \mathcal{F}$.

Proposition 2.3.1 *Let* $(s,t) \in \mathcal{R}$ *and* $\tau(s,t) = y^i$. *If* $S_j \neq T_j$ *for some* $j \neq i$, *then* $(y^i, y^j) \in \mathcal{F}$.

Proof: Since $j \neq i$, we have $s_j = t_j$. Thus, if $S_j \neq T_j$, the value of some input argument of Δ_j must have changed along with the state transition. As the only state variable changed is y^i, by Definitions 2.3.1 and 2.3.6, $(y^i, y^j) \in \mathcal{F}$. □

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

$$\rho(s, y^i) = \{t \in \mathcal{Q} \mid s\mathcal{R}t \text{ and } \tau(s,t) = y^i\}$$

  for $s \in \mathcal{Q}$ and $y^i \in \mathcal{Y}$;
- $\mathcal{Q}$ is the set of accepting states.

The set of all words over the alphabet $\mathcal{Y}$ is denoted by $\mathcal{Y}^*$. We extend the state transition function ρ to $\rho^* : \mathcal{Q} \times \mathcal{Y}^* \to 2^{\mathcal{Q}}$, which can be defined inductively as follows. For $s \in \mathcal{Q}$, $y^i \in \mathcal{Y}$, and $w \in \mathcal{Y}^*$,

- $\rho^*(s, \epsilon) = \{s\}$;
- $\rho^*(s, wy^i) = \{t \in \mathcal{Q} \mid t \in \rho(s', y^i) \text{ for some } s' \in \rho^*(s, w)\}$.

We often represent the state set $\rho^*(s, w)$ 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. That is, if $w \in L(\mathcal{B})$, then $w' \in L(\mathcal{B})$ for any prefix w' of w. 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. A trace set is not quite as descriptive as the graph B itself, in terms of its ability to characterize the exact behavior of a network. Because of nondeterminism, a string in $L(\mathcal{B})$ does not necessarily correspond to a unique reachable state.

Example 2.3.3 Consider the network behavior in Figure 2.4. The corresponding trace set $\mathcal{T}$ is defined as follows:

$$\mathcal{T} = \{w \in \mathcal{Y}^* \mid w \text{ is a prefix of } w' \text{ for some } w' \in (y^2y^3y^1)^*\}.$$

For the behavior in Figure 2.5. the corresponding trace set is

$$\{w \in \mathcal{Y}^* \mid w \text{ is a prefix of } w' \text{ for some } w' \in ((y^2y^3 + y^3y^2)y^1)^*\}.$$

Chapter 3

Delay-Insensitivity

In this chapter, we first describe how we model all possible delay distributions in an asynchronous network. We then give a definition of delay-insensitivity of networks using the notion of behavioral equivalence in the presence of arbitrary delays.

3.1 Delay Extensions

To determine whether a network is delay-insensitive, we wish to examine how the distribution of delays in the modules and in the connections affects the behavior of the network. The motivation behind our approach is Charles Molnar's "foam-rubber wrapper" principle [13], 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. This intuitive idea corresponds to surrounding each module in a network by DELAY modules. However, instead of "wrapping" all the modules by DELAY modules at once, we choose to proceed inductively using the notion of *delay extension*.

Definition 3.1.1 Let N be a network. A *delay successor* of N is a network $\hat{N}$ obtained from N by inserting a DELAY module M^{n+1} into some connection $e = (z_k^i, x_l^j)$ of N. Formally, we have $\hat{N} = \langle \hat{\mathcal{M}}, \hat{G} \rangle$, where

- $\hat{\mathcal{M}} = \mathcal{M} \cup \{M^{n+1}\}$;
- $\hat{G} = \langle \hat{\mathcal{V}}, \hat{\mathcal{E}} \rangle$, where
  - $\hat{\mathcal{V}} = \mathcal{V} \cup \mathcal{V}^{n+1}$;
  - $\hat{\mathcal{E}} = (\mathcal{E} - \{e\}) \cup \mathcal{E}^{n+1} \cup \{(z_k^i, x_1^{n+1}), (z_1^{n+1}, x_l^j)\}$.

Definition 3.1.2 The set $\mathcal{D}(N)$ of *delay extensions* of a network N is defined inductively, as follows,

- $N \in \mathcal{D}(N)$;
- if $\hat{N} \in \mathcal{D}(N)$, then each delay successor of $\hat{N}$ is also in $\mathcal{D}(N)$.

As a convention, we denote a delay extension of a network N by $\hat{N}$. Furthermore, all the components of $\hat{N}$, any object associated with $\hat{N}$, e.g., a behavior, and the components of that object are all marked by $\hat{}$.

In order to compare the behavior of a network N with the behavior of a delay extension of N, it is essential that we relate the corresponding states and state transition sequences in the two behaviors.

Definition 3.1.3 Let $\hat{N} \in \mathcal{D}(N)$. The *projection of a state* $\hat{s}$ of $\hat{N}$ with respect to N is the $|\mathcal{Y}|$-tuple $s = \hat{s} \downarrow \mathcal{Y}$ obtained from $\hat{s}$ by removing all components corresponding to variables not in $\mathcal{Y}$. The *projection of a word* $\hat{w} \in \hat{\mathcal{Y}}^*$ with respect to N is the word $w = \hat{w} \downarrow \mathcal{Y}$ obtained from $\hat{w}$ by erasing all the symbols not in $\mathcal{Y}$.

Definition 3.1.4 Let $\hat{N} \in \mathcal{D}(N)$, and let s be a state of N. An *extension* of s with respect to $\hat{N}$ is a state $\hat{s}$ of $\hat{N}$ for which $\hat{s} \downarrow \mathcal{Y} = s$. Let $\hat{s}$ be an extension of s; if $\mathcal{U}(\hat{s}) \subseteq \mathcal{Y}$, in other words, if all the inserted DELAY modules are stable in $\hat{s}$, then we say that $\hat{s}$ is the *stable-delay extension* of s.

We now prove two useful propositions concerning state extensions. In the following, $\hat{N} \in \mathcal{D}(N)$, and s and $\hat{s}$ are states of N and $\hat{N}$, respectively. We call a module of $\hat{N}$ which is also present in N an *original* module.

Proposition 3.1.1 *If $\hat{s}$ is the stable-delay extension of s, then $S_i = \hat{S}_i$ for all i such that M^i is an original module.*

Proof: Let M^i be original. If an inserted DELAY module M^d feeds M^i in $\hat{N}$, then there must be an original module M^h which feeds the head of a chain of inserted DELAY modules ending at M^d, and $(M^h, M^i) \in \mathcal{F}$. Since all the inserted DELAY modules are stable in $\hat{s}$, their values agree with their inputs. It follows that the input states of M^i are identical in the two states s and $\hat{s}$. Also, $s_i = \hat{s}_i$. Therefore, $S_i = \hat{S}_i$. □

Corollary If $\hat{s}$ is the stable-delay extension of s, then $\mathcal{U}(\hat{s}) = \mathcal{U}(s)$.

Proposition 3.1.2 *Let $\hat{s}$ be an extension of s. Then there exists $\hat{w} \in (\hat{\mathcal{Y}} - \mathcal{Y})^*$ such that $\hat{s}\hat{w} = \{\hat{s}'\}$, where $\hat{s}'$ is the stable-delay extension of s.*

Proof: Let us assign "levels" to the *inserted* DELAY modules in $\hat{N}$ inductively as follows: a) A DELAY module is of level 1 if it is fed by an original module; b) a DELAY module is of level $k+1$ if it is fed by a DELAY module of level k. Clearly, there are no cycles of inserted DELAY modules in $\hat{N}$. Also, every DELAY module has a unique input and output. Therefore, each inserted DELAY module

is assigned a unique level. Starting in state $\hat{s}$, we change inserted DELAY modules which are unstable in such a way that at each step, we always choose one which has the least level. We stop when all the inserted DELAY modules are stabilized. One easily verifies that the process will terminate when the stable-delay extension $\hat{s}'$ of s is reached. Moreover, since DELAY modules are deterministic, $\hat{s}'$ is the only state reachable. □

3.2 Delay-Insensitivity of Networks

The delay extensions of a network provide a description of all possible delay distributions within the network. For a network to be delay-insensitive, the behavior of any delay extension of the network should satisfy a certain correctness concern with respect to the behavior of the original network. We define this correctness concern to correspond to *weak bisimulation* [12] (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.

In order to compare the behavior of a network with the behavior of a delay extension of the network, we have to choose an initial state for the latter behavior. It only makes sense to choose an initial state which "imitates" the initial state of the behavior of the original network.

Definition 3.2.1 Let $\hat{N} \in \mathcal{D}(N)$, and 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 the stable-delay extension of q.

We now define delay-insensitivity of networks formally. Note that in addition to observation equivalence, we require *livelock-freedom*.

Definition 3.2.2 Let $\hat{N} \in \mathcal{D}(N)$. Let $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ be a behavior of N, and let $\hat{B} = \langle \hat{q}, \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ be the behavior of $\hat{N}$ which is initial-state compatible with B.

- The behavior $\hat{B}$ is said to be *delay-safe with respect to* B if, whenever $\hat{s} \in \hat{\mathcal{Q}}$ is an extension of $s \in \mathcal{Q}$, then for all $\hat{t} \in \hat{\mathcal{Q}}$ and $\hat{w} \in \hat{\mathcal{Y}}^*$, if $\hat{t} \in \hat{s}\hat{w}$ and $\hat{w} \downarrow \mathcal{Y} = y^i$ for some $y^i \in \mathcal{Y}$, then there exists $t \in \mathcal{Q}$ such that $t \in sy^i$ and t is the projection of $\hat{t}$ with respect to N.

- The behavior $\hat{B}$ is said to be *delay-faithful with respect to* B if, whenever $\hat{s} \in \hat{\mathcal{Q}}$ is an extension of $s \in \mathcal{Q}$, then for all $t \in sy^i$, there exist $\hat{w} \in \hat{\mathcal{Y}}^*$ and $\hat{t} \in \hat{\mathcal{Q}}$ such that $\hat{t} \in \hat{s}\hat{w}$, $\hat{w} \downarrow \mathcal{Y} = y^i$, and $\hat{t}$ is an extension of t with respect to $\hat{N}$.

- The behavior $\hat{B}$ is said to be *livelock-free with respect to* B if, for all $\hat{s} \in \hat{\mathcal{Q}}$ and $\hat{w} \in (\hat{\mathcal{Y}} - \mathcal{Y})^+$, $\hat{s} \notin \hat{s}\hat{w}$.

Definition 3.2.3 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, where $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ is the behavior of N originating from q, and $\hat{B}$ is the behavior of $\hat{N}$ which is initial-state compatible with B.

Delay-safety and delay-faithfulness constitute the bisimulative relationship between the behaviors B and $\hat{B}$. Livelock-freedom requires that no state of $\hat{N}$ is reachable from itself via a sequence of transitions on the inserted DELAY modules only.

We now show that delay-faithfulness and livelock-freedom are redundant properties, since they are satisfied by all delay extensions. Consequently, only delay-safety is required for delay-insensitivity.

Proposition 3.2.1 *The behavior $\hat{B}$ of Definition 3.2.2 is always livelock-free with respect to B.*

Proof: Suppose that there exist $\hat{s} \in \hat{\mathcal{Q}}$ and $\hat{w} \in (\hat{\mathcal{Y}} - \mathcal{Y})^+$ such that $\hat{s} \in \hat{s}\hat{w}$. Let y^i be an inserted DELAY module which appears in $\hat{w}$. Clearly, y^i must appear at least twice in $\hat{w}$. Thus, the (unique) state variable y^h that feeds y^i must have changed and appears in $\hat{w}$. It follows that y^h is an inserted DELAY module. Applying the same argument (this time, on y^h, to start with) repeatedly will lead to a contradiction since there are no cycles of inserted DELAY modules in $\hat{N}$. We are done. □

Proposition 3.2.2 *The behavior $\hat{B}$ of Definition 3.2.2 is always delay-faithful with respect to B.*

Proof: Let $\hat{s} \in \hat{\mathcal{Q}}$ be an extension of $s \in \mathcal{Q}$, and let $t \in sy^i$. By Proposition 3.1.2, there exists $\hat{w} \in (\hat{\mathcal{Y}} - \mathcal{Y})^*$ such that $\hat{s}\hat{w} = \{\hat{s}'\}$, where $\hat{s}'$ is the stable-delay extension of s. By Proposition 3.1.1, $S_i = \hat{S}'_i$. Therefore, there exists $\hat{t} \in \hat{s}'y^i$ such that $\hat{t}$ is an extension of t with respect to N. Thus, we have $\hat{t} \in \hat{s}'y^i \subseteq \hat{s}\hat{w}y^i$, $\hat{w}y^i \downarrow \mathcal{Y} = y^i$, and $\hat{t}$ is an extension of t with respect to N. By Definition 3.2.2, $\hat{B}$ is delay-faithful with respect to B. □

In view of Propositions 3.2.1 and 3.2.2, we have the following:

Theorem 3.1 *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 with respect to B, where B and $\hat{B}$ are as defined in Definition 3.2.3.*

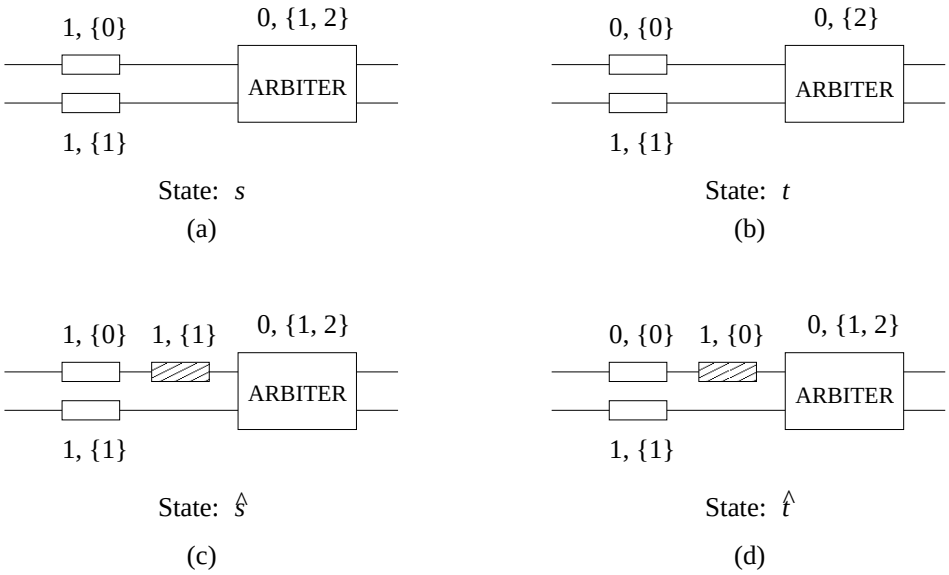


Figure 3.1: Example of delay-safety violation.

Example 3.2.1 Consider part of a network N shown in Figure 3.1(a) (for brevity, we hide the internal structures of modules), where two DELAY modules are connected to an ARBITER module (as defined in Example 2.1.8). Figure 3.1(c) shows the corresponding part of a delay successor $\hat{N}$ of N, 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, as shown in Figure 3.1(b), results from state s (Figure 3.1(a)) by a change on the unstable DELAY module. State $\hat{t}$ of $\hat{N}$ (Figure 3.1(d)) results from state $\hat{s}$ (Figure 3.1(c)) by the same change; also, $\hat{s}$ is the stable-delay extension of s. We assume that the states of network modules not shown are the same in all four states.

Clearly, $\hat{t}$ is an extension of t. 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 is not delay-insensitive with respect to s. □

Chapter 4

Semi-Modularity

In this chapter, we first generalize Muller's definition of semi-modularity to permit us to include nondeterministic modules. We then define a special class of networks, the delay-dense networks, which characterize asynchronous circuits properly for the purpose of delay-insensitivity analysis. We prove that for delay-dense networks, semi-modularity is preserved under delay extensions. Finally, we concentrate on the relationship between semi-modularity and Udding's definition of delay-insensitivity [19]. We prove that 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 the JTU-Rules.

4.1 Generalization of Semi-Modularity

Semi-modularity of circuit behaviors was initially defined in [11] 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 cor-

responding to such a behavior was shown to be a semi-modular lattice. In this section, we generalize this original definition of semi-modularity to suit our network model, where state variables may have several implied values at a time (e.g., an ARBITER module), causing nondeterminism in network behaviors. However, we continue to use the term "semi-modular" even though the same partially ordered set corresponding to a nondeterministic "semi-modular" behavior is no longer a semi-modular lattice.

Definition 4.1.1 Let $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ be a behavior of a network N. A state $s \in \mathcal{Q}$ is said to be *semi-modular* if, for all $t \in \mathcal{Q}$, if $(s,t) \in \mathcal{R}$ with $\tau(s,t) = y^i$, then $S_j \subseteq T_j$ for all $j \neq i$ such that $y^j \in \mathcal{U}(s)$.

Definition 4.1.2 A behavior $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ of a network N is *semi-modular* if, for all $s \in \mathcal{Q}$, s is semi-modular.

Intuitively, semi-modularity requires that once a state variable y^i becomes unstable and b is in its excitation, b should remain in the excitation until y^i changes; in other words, an enabled action on a state variable cannot be disabled by an action on another state variable.

Example 4.1.1 For a violation of semi-modularity, refer to Figures 3.1(a) and (b). The change on the DELAY module involved alters the excitation of the ARBITER module and violates semi-modularity.

Example 4.1.2 Figure 4.1 shows a network N_3 consisting of an ARBITER module, an INVERTED C-ELEMENT module, and two DELAY modules. An INVERTED C-ELEMENT module functions like a Muller C-ELEMENT module with an inverted output. Figure 4.2 shows a semi-modular behavior of N_3. This example illustrates

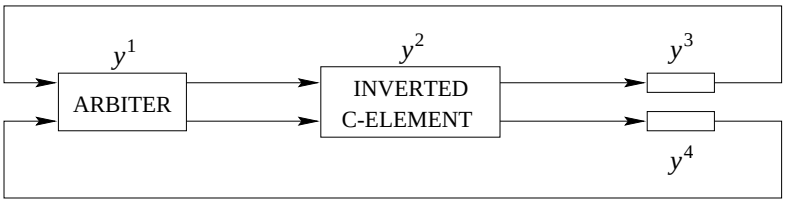


Figure 4.1: Network example: N_3.

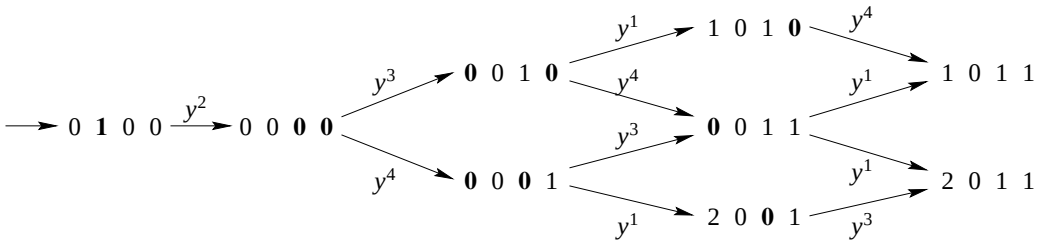


Figure 4.2: A semi-modular behavior of network N_3.

that the set inclusion in Definition 4.1.1 can be proper. For example, in state **0**010, the ARBITER module can change to state 1 only; in a subsequent state **0**011, resulting from a change on a DELAY module, the ARBITER module can change to either 1 or 2.

Proposition 4.1.1 *A behavior $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ is semi-modular if and only if, whenever $s, t \in \mathcal{Q}$, $t \in sw$ for some $w \in \mathcal{Y}^*$, $y^i \in \mathcal{U}(s)$, and $S_i \not\subseteq T_i$, then y^i appears in w.*

Proof: We prove the "if" part by contradiction. Assume that B is not semi-modular. Then there exists $(s, t) \in \mathcal{R}$ with $\tau(s, t) = y^j$ such that $S_i \not\subseteq T_i$ for some

$i \neq j$ with $y^i \in \mathcal{U}(s)$. Let $w = y^j$. Then we have $t \in sw$, $S_i \not\subseteq T_i$, but y^i does *not* appear in w. Now assume that B is semi-modular. Let $s, t \in \mathcal{Q}$ be such that $t \in sw$ for some $w \in \mathcal{Y}^*$ and $S_i \not\subseteq T_i$. Clearly, $w \neq \epsilon$. If y^i does not appear in w, then by semi-modularity and transitivity of set inclusions, we would have $S_i \subseteq T_i$, which is a contradiction. Hence, the proposition is proved. □

4.2 Semi-Modularity and Delay-Density

4.2.1 Delay-Dense Networks

An asynchronous circuit consists of a set of components interconnected by wires, where each component can be modelled by a module. The distinction between the current state and the current excitation of a module allows us to associate a delay with the module. We model wire delays by treating wires as 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, resulting in a *delay-dense* network, defined as follows:

Definition 4.2.1 A network N, as in Definition 2.2.1, is *delay-dense* if, whenever $(M^i, M^j) \in \mathcal{F}$, then either M^i or M^j is a DELAY module.

Thus, in a delay-dense network, of each pair of adjacent modules in the network, at least one is a DELAY module. For example, the networks of Examples 2.2.1 and 2.2.2 are not delay-dense. They become delay-dense if we insert a DELAY module in each of the network connections.

Proposition 4.2.1 *If N is delay-dense and $(M^i, M^i) \in \mathcal{F}$ for some M^i, then N consists of a single* DELAY *module.*

Proof: If N is delay-dense and $(M^i, M^i) \in \mathcal{F}$, then M^i must be a DELAY module and $(z_1^i, x_1^i) \in \mathcal{K}$. Thus, the vertices of M^i cannot be connected with vertices of other modules. Since the network graph is connected, N consists of a single DELAY module, namely, M^i. □

4.2.2 Semi-Modular Delay-Dense Networks

We now prove that for delay-dense networks, semi-modularity is preserved under delay extensions. We first prove that semi-modularity of behaviors of delay-dense networks is preserved through the addition of one DELAY module. We then extend this result to arbitrary delay extensions.

Theorem 4.1 *Let $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ be a semi-modular behavior of a network N. Let $\hat{N}$ be a delay successor of N, where the extra* DELAY *module M^d is inserted in a connection between two distinct modules. Without loss of generality, assume that the connection is from M^1 to M^2. If M^2 is a* DELAY *module, then $\hat{B} = \langle \hat{q}, \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$, the behavior of $\hat{N}$ which is initial-state compatible with B, is also semi-modular.*

Proof: By the construction of $\hat{N}$, we have $(y^1, y^2) \in \mathcal{F}$ and $(y^1, y^d), (y^d, y^2) \in \hat{\mathcal{F}}$. The result follows from a series of four lemmata. In the following, we assume that $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ is semi-modular. Also, let $\hat{s} \in \hat{\mathcal{Q}}$ be any state of $\hat{N}$. The state $\hat{s}$ is said to be *normal* if $\hat{s} \downarrow \mathcal{Y} \in \mathcal{Q}$; in that case, we let $s = \hat{s} \downarrow \mathcal{Y}$.

Lemma 4.1.1 *Let $\hat{s}$ be normal. If $i \neq d$ and $i \neq 2$, then $\hat{S}_i = S_i$ and $\mathcal{U}(\hat{s}) - \{y^d, y^2\} \subseteq \mathcal{U}(s)$.*

Proof: Clearly, M^i is an original module. One easily verifies that the input connections to M^i are identical in the two networks N and $\hat{N}$. Thus, the value of each argument of Δ_i in s is equal to the value of the corresponding argument of $\hat{\Delta}_i$ in $\hat{s}$. Hence, $\hat{S}_i = S_i$. The second result follows immediately. □

Lemma 4.1.2 *If $\hat{s}$ is normal, but not semi-modular, then $y^d \in \mathcal{U}(\hat{s})$.*

Proof: Let $s = \hat{s} \downarrow \mathcal{Y} \in \mathcal{Q}$. Suppose that $\hat{s}$ is not semi-modular. Then there exists $\hat{t} \in \hat{\mathcal{Q}}$ such that

1. $(\hat{s}, \hat{t}) \in \hat{\mathcal{R}}$ with $\tau(\hat{s}, \hat{t}) = y^i$;
2. there exists $j \neq i$ such that $y^j \in \mathcal{U}(\hat{s})$ and $\hat{S}_j \not\subseteq \hat{T}_j$.

By Proposition 2.3.1, $(y^i, y^j) \in \hat{\mathcal{F}}$. Also, $y^i \in \mathcal{U}(\hat{s})$, by definition. Assume that $y^d \notin \mathcal{U}(\hat{s})$. Then, $i \neq d$ and $j \neq d$. Since M^2 (a DELAY module) has exactly one input, $i \neq d$, and (y^i, y^j), $(y^d, y^2) \in \hat{\mathcal{F}}$, we must have $j \neq 2$.

Since $y^d \notin \mathcal{U}(\hat{s})$, $\hat{s}$ is the stable-delay extension of s. Also, M^i and M^j are original. By Proposition 3.1.1, $\hat{S}_j = S_j$ and $\hat{S}_i = S_i$. It follows that there exists $t \in \mathcal{Q}$ such that $s\mathcal{R}t$, $\tau(s, t) = y^i$, and $t = \hat{t} \downarrow \mathcal{Y}$. Thus, $\hat{t}$ is normal. Recall that $j \neq d$ and $j \neq 2$. By Lemma 4.1.1, $\hat{T}_j = T_j$. On the other hand, $y^j \in \mathcal{U}(\hat{s}) = \mathcal{U}(s)$, by the corollary of Proposition 3.1.1. Also, B is semi-modular and $j \neq i$. Therefore, $S_j \subseteq T_j$. It follows that $\hat{S}_j = S_j \subseteq T_j = \hat{T}_j$—a contradiction. Hence, if $\hat{s}$ is not semi-modular, then $y^d \in \mathcal{U}(\hat{s})$. □

Lemma 4.1.3 *If $\hat{s}$ is normal, but not semi-modular, then either $y^2 \in \mathcal{U}(\hat{s})$ or there exists $\hat{t} \in \hat{\mathcal{Q}}$ such that $(\hat{s}, \hat{t}) \in \hat{\mathcal{R}}$, $\tau(\hat{s}, \hat{t}) = y^1$, and $\hat{S}_d \not\subseteq \hat{T}_d$.*

Proof: Let $s = \hat{s} \downarrow \mathcal{Y} \in \mathcal{Q}$. Suppose that $\hat{s}$ is not semi-modular. Let i, j, and $\hat{t}$ be as defined in the proof of Lemma 4.1.2. Again, we have $(y^i, y^j) \in \hat{\mathcal{F}}$ and $y^i \in \mathcal{U}(\hat{s})$.

Assume that $y^2 \notin \mathcal{U}(\hat{s})$. Then, $i \neq 2$ and $j \neq 2$. Since M^d (a DELAY module) has exactly one output, $j \neq 2$, and (y^i, y^j), $(y^d, y^2) \in \hat{\mathcal{F}}$, we must have $i \neq d$.

Suppose that $j \neq d$. By Lemma 4.1.1, $\hat{S}_j = S_j$. Similarly, $\hat{S}_i = S_i$. It follows that there exists $t \in \mathcal{Q}$ such that $s\mathcal{R}t$, $\tau(s,t) = y^i$, and $t = \hat{t} \downarrow \mathcal{Y}$. Thus, $\hat{t}$ is normal. By Lemma 4.1.1 again, $\hat{T}_j = T_j$. On the other hand, by Lemma 4.1.1, $y^j \in \mathcal{U}(\hat{s}) - \{y^d, y^2\} \subseteq \mathcal{U}(s)$. Also, B is semi-modular and $j \neq i$. Therefore, $S_j \subseteq T_j$. It follows that $\hat{S}_j = S_j \subseteq T_j = \hat{T}_j$—a contradiction. Hence, $j = d$. Since $(y^i, y^j) \in \hat{\mathcal{F}}$, we must have $i = 1$. Thus, $(\hat{s}, \hat{t}) \in \hat{\mathcal{R}}$, $\tau(\hat{s}, \hat{t}) = y^1$, and $\hat{S}_d \not\subseteq \hat{T}_d$, as in the lemma. □

Lemma 4.1.4 *For all $\hat{s} \in \hat{\mathcal{Q}}$, $\hat{s}$ is semi-modular and normal.*

Proof: We proceed by induction. For the base case, let $\hat{s} = \hat{q}$. Clearly, $\hat{s}$ is normal and $y^d \notin \mathcal{U}(\hat{s})$. By Lemma 4.1.2, $\hat{s}$ is semi-modular. Now, let $(\hat{r}, \hat{s}) \in \hat{\mathcal{R}}$ with $\tau(\hat{r}, \hat{s}) = y^i$. Assume that $\hat{r}$ is semi-modular and normal. We shall prove that $\hat{s}$ is semi-modular and normal.

Let $r = \hat{r} \downarrow \mathcal{Y} \in \mathcal{Q}$. If $y^d \notin \mathcal{U}(\hat{r})$, then $i \neq d$ and $\hat{R}_i = R_i$. It follows that $\hat{s} \downarrow \mathcal{Y} \in \mathcal{Q}$. Now assume that $y^d \in \mathcal{U}(\hat{r})$. If $y^2 \in \mathcal{U}(\hat{r})$, then by changing y^d in state $\hat{r}$, we would have a violation of semi-modularity, contradicting the semi-modularity of $\hat{r}$. Therefore, $y^2 \notin \mathcal{U}(\hat{r})$ and $i \neq 2$. If $i = d$, then $\hat{s} \downarrow \mathcal{Y} = \hat{r} \downarrow \mathcal{Y} \in \mathcal{Q}$; otherwise, by Lemma 4.1.1, we have $\hat{R}_i = R_i$, which implies that $\hat{s} \downarrow \mathcal{Y} \in \mathcal{Q}$. Hence, $\hat{s}$ is normal.

Suppose that $\hat{s}$ is not semi-modular. Let the output vertex of M^1 which is connected to M^d be z_k^1. Clearly, z_k^1 is connected to M^2 in N. By Lemma 4.1.2, $y^d \in \mathcal{U}(\hat{s})$. It follows that $i \neq d$. By Lemma 4.1.3, there are two cases to consider.

- $y^2 \in \mathcal{U}(\hat{s})$:

Note that y^2 is a DELAY module. Since $i \neq d$, $y^2 \in \mathcal{U}(\hat{r})$, by Proposition 2.3.1. By semi-modularity of $\hat{r}$, we have $y^d \notin \mathcal{U}(\hat{r})$. Thus, $i = 1$ and $y^1 \in \mathcal{U}(\hat{r})$. On the other hand, since $y^d \notin \mathcal{U}(\hat{r})$, $\hat{r}$ is the stable-delay extension of r. It follows that $y^1, y^2 \in \mathcal{U}(\hat{r}) = \mathcal{U}(r)$ and there exists $s \in \mathcal{Q}$ such that $r\mathcal{R}s$, $\tau(r,s) = y^1$, and $s = \hat{s} \downarrow \mathcal{Y}$. We have $R_2 = \lambda_k^1(r_1) = \lambda_k^1(\hat{r}_1) = \hat{R}_d = \hat{r}_d$ and $S_2 = \lambda_k^1(s_1) = \lambda_k^1(\hat{s}_1) = \hat{S}_d$. Since $y^d \in \mathcal{U}(\hat{s})$, $\hat{s}_d \neq \hat{S}_d$. But $\hat{r}_d = \hat{s}_d$. Therefore, $R_2 \neq S_2$, which violates semi-modularity of B—a contradiction.

- There exists $\hat{t} \in \hat{\mathcal{Q}}$ such that $(\hat{s}, \hat{t}) \in \hat{\mathcal{R}}$, $\tau(\hat{s}, \hat{t}) = y^1$, and $\hat{S}_d \not\subseteq \hat{T}_d$:

  Since $y^2 \notin \mathcal{U}(\hat{s})$, $\hat{s}_2 = \hat{s}_d$. Also, $\hat{s}_d \neq \lambda_k^1(\hat{s}_1)$, since $y^d \in \mathcal{U}(\hat{s})$. Let $s = \hat{s} \downarrow \mathcal{Y} \in \mathcal{Q}$. Then, $s_2 \neq \lambda_k^1(s_1)$, which implies that $y^2 \in \mathcal{U}(s)$. On the other hand, by Lemma 4.1.1, we have $\hat{S}_1 = S_1$. Therefore, there exists $t \in \mathcal{Q}$ such that $s\mathcal{R}t$, $\tau(s,t) = y^1$, and $t = \hat{t} \downarrow \mathcal{Y}$. We have $S_2 = \lambda_k^1(s_1) = \lambda_k^1(\hat{s}_1) = \hat{S}_d$, and similarly, $T_2 = \hat{T}_d$. Since $\hat{S}_d \not\subseteq \hat{T}_d$, i.e., $\hat{S}_d \neq \hat{T}_d$, we have $S_2 \neq T_2$, which violates semi-modularity of B—a contradiction again.

Hence, $\hat{s}$ is semi-modular. The lemma is proved by induction. □

By Lemma 4.1.4, $\hat{s}$ is semi-modular for all $\hat{s} \in \hat{\mathcal{Q}}$. Hence, $\hat{B}$ is semi-modular. We are done. □

Theorem 4.2 *Let N be a delay-dense network, and let $\hat{N} \in \mathcal{D}(N)$. Let $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ be a behavior of N, and let $\hat{B} = \langle \hat{q}, \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ be the behavior of $\hat{N}$ which is initial-state compatible with B. If B is semi-modular, then so is $\hat{B}$.*

Proof: We prove the theorem by induction on the number of inserted DELAY modules. The base case, where $\hat{N} = N$, is trivial. Now assume that $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ is semi-modular. We insert a DELAY module M^d into a connection e of N. If

e connects a module to itself, then by Proposition 4.2.1, N consists of a single DELAY module; the result holds trivially in that case. Thus we assume otherwise, i.e., M^d is inserted between two distinct modules M^i and M^j. Denote the resulting delay successor by $\hat{N}$. Thus, (M^i, M^d), $(M^d, M^j) \in \hat{\mathcal{F}}$. Let $\hat{B} = \langle \hat{q}, \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ be the behavior of $\hat{N}$ which is initial-state compatible with B. We shall prove that $\hat{B}$ is semi-modular.

If M^j is a DELAY module, then the result follows from Theorem 4.1; otherwise, since N is delay-dense, M^i must be a DELAY module. Let $(M^h, M^i) \in \mathcal{F}$. Clearly, M^h is unique. Imagine that in the network N, we change the index of M^i to d, and call the new network $\tilde{N}$. Obviously, B is a semi-modular behavior of $\tilde{N}$ as well. Now $\hat{N}$ can be viewed as being obtained from $\tilde{N}$ by inserting M^i in the connection between M^h and M^d, in which case the new DELAY module feeds the DELAY module M^d. So the result we wish to prove again follows from Theorem 4.1. The theorem is proved by induction. □

It is worth noting that Theorem 4.2 does not necessarily hold for networks which are not delay-dense. For example, consider the network consisting of a two-output INVERTER module and an AND-gate, as shown in Figure 4.3(a). Let the initial state be **0**0. The corresponding behavior is semi-modular, as shown in Figure 4.3(b). With a DELAY module y^3 inserted, as shown in Figure 4.3(a), the corresponding behavior (Figure 4.3(c)) is no longer semi-modular; more specifically, the states **10**0 and **00**1 are not semi-modular.

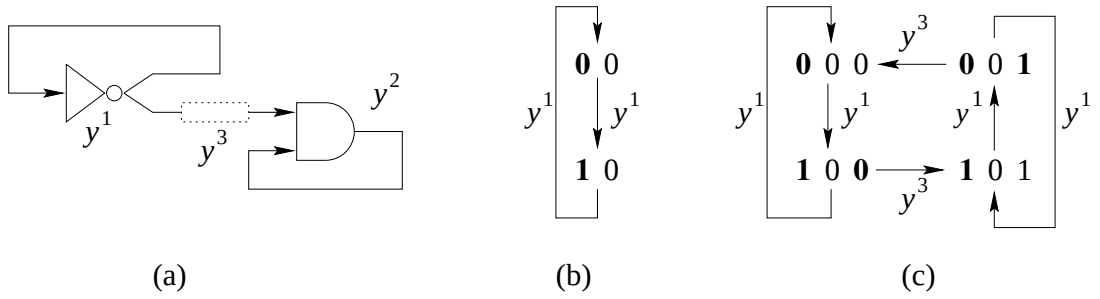


Figure 4.3: Theorem 4.2 fails to hold for networks which are not delay-dense.

4.3 Semi-Modularity and the JTU-Rules

In this section, we investigate the relationship between semi-modularity and Udding's definition of delay-insensitivity. We are only concerned with delay-dense networks. The reasoning behind this will become clear later. For simplicity, we disregard the trivial network consisting of a single DELAY module. Thus, all networks under discussion are free of self-loops. Also, we assume that all modules are one-step. We first prove several results concerning the ordering among state transitions in a semi-modular behavior. We then describe the JTU-Rules and Udding's definition of delay-insensitivity. Finally, we prove that the semi-modularity of a network's behavior implies the delay-insensitivity of its components, according to Udding's criterion.

The notation used in this section is somewhat different from that in the other sections. Instead of using y^i's, we denote state variables by letters at the beginning of the Latin alphabet up to g. Variable names are also used as indices. For example, the excitation of variable a in state s is denoted by S_a. Strings over the alphabet $\mathcal{Y}$ are denoted by letters at the end of the Latin alphabet, starting with u. Occurrences

of the same symbol in a string are ordered with the first occurrence having index 1. Superscripts are used exclusively to indicate occurrence indices. For example, the string $u = abaaba$ implicitly represents $a^1b^1a^2a^3b^2a^4$. If a symbol a occurs in a string y, we write $a \in y$; otherwise, we write $a \notin y$.

4.3.1 The "Chain" Theorem

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. Recall that the language $L(\mathcal{B})$ accepted by $\mathcal{B}$ is defined as

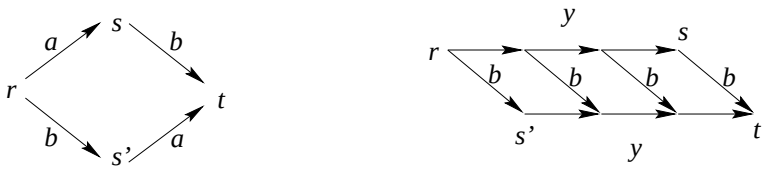
$$L(\mathcal{B}) = \{w \in \mathcal{Y}^* \mid qw \neq \emptyset\}.$$

Definition 4.3.1 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 $a^i \rightsquigarrow_u b^j$, if and only if

1. $(a, b) \in \mathcal{F}$, and
2. $a, b \notin y$.

For example, consider the behavior in Figure 2.5. Let $u = y^3y^2y^1y^2y^3y^1$. Then we have $y^{3,1} \rightsquigarrow_u y^{1,1}$, $y^{2,1} \rightsquigarrow_u y^{1,1}$, and $y^{1,1} \rightsquigarrow_u y^{3,2}$ among others. If u is implied by the context, we often omit u from $\rightsquigarrow_u$. The reflexive and transitive closure $\rightsquigarrow^*$ of $\rightsquigarrow$ can be defined inductively as follows,

- $a^i \rightsquigarrow^* a^i$;
- if $a^i \rightsquigarrow b^j$ and $b^j \rightsquigarrow^* c^k$, then $a^i \rightsquigarrow^* c^k$.



(a) (b)

Figure 4.4: Pictorial proof of Proposition 4.3.2 and its corollary.

Proposition 4.3.1 *For any $u \in L(\mathcal{B})$, no two consecutive symbols in u are the same.*

Proof: This is because all modules are one-step and N contains no self-loops. □

Proposition 4.3.2 *Let $r, t \in \mathcal{Q}$ and $a, b \in \mathcal{Y}$ be such that $t \in rab$. Then either $(a, b) \in \mathcal{F}$ or $t \in rba$.*

Proof: By Proposition 4.3.1, $a \neq b$. Let $r \xrightarrow{a} s \xrightarrow{b} t$ be a path in B, as shown in Figure 4.4(a), where $s \in \mathcal{Q}$. If $(a, b) \notin \mathcal{F}$, then by Proposition 3.1, $R_b = S_b$. Thus, we have $r \xrightarrow{b} s'$ for some $s' \in \mathcal{Q}$ such that $s'_b = t_b$. Note that B is semi-modular and $a \in \mathcal{U}(r)$. Therefore, $R_a \subseteq S'_a$, 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., $t \in rba$. □

Corollary Let $r, t \in \mathcal{Q}$, $b \in \mathcal{Y}$, and $y \in \mathcal{Y}^*$ be such that $t \in ryb$. If for all symbols a in y, $(a, b) \notin \mathcal{F}$, then $t \in rby$.

Proof: This is obvious by Proposition 4.3.2, as shown in Figure 4.4(b). □

Definition 4.3.2 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.

Theorem 4.3 ("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 $y'y'' \sim y$.*

Proof: Let us prove the theorem by induction on $|y|$. The base case, where $|y| = 0$, follows directly from Proposition 4.3.2. Now assume that the theorem holds for $0 \leq |y| \leq N$. Let $u = xa^iyb^jz$, where $|y| = N + 1$. We now prove the following claim.

Lemma 4.3.1 *If $w = x'c^ky'c^lz' \in L(\mathcal{B})$ for some $c \in \mathcal{Y}$ and $x', y', z' \in \mathcal{Y}^*$, and $|y'| \leq N$, then $c^k \rightsquigarrow_w^* c^l$.*

Proof: If $c \notin y'$, then by the induction hypothesis and Proposition 4.3.1, we must have $c^k \rightsquigarrow_w^* c^l$. If $c \in y'$, then we can simply consider each interval of $c^ky'c^l$ delimited by c separately. □

Returning to the proof of the theorem, assume that $a^i \not\rightsquigarrow_u^* b^j$. Let d^l be the *leftmost* symbol in yb^j for which $d^l \rightsquigarrow_u^* b^j$. Clearly, such d^l exists, as $b^j \rightsquigarrow_u^* b^j$. Rewrite u as $u = xa^iy'd^ly''z$, where $0 \leq |y'|, |y''| \leq N + 1$. Thus, for all $c^k \in a^iy'$, $c^k \not\rightsquigarrow_u d^l$. Our next result shows that d^l can be "moved" in front of a^i.

Lemma 4.3.2 $xd^la^iy'y''z \in L(\mathcal{B})$.

Proof: If for all symbols c in a^iy', we have $(c, d) \notin \mathcal{F}$, then the lemma holds, by the corollary of Proposition 4.3.2. Otherwise, let c^k be the rightmost symbol in a^iy' for which $(c, d) \in \mathcal{F}$. Write $a^iy'd^l$ as $y_1'c^ky_2'd^l$. Clearly, $c \notin y_2'$. Note that $|y_2'| \leq N + 1$. If $d \in y_2'$, then by Lemma 4.3.1, we would contradict our choice of d^l. If $d \notin y_2'$, then by definition, $c^k \rightsquigarrow_u d^l$, a contradiction again. Hence, the lemma is proved. □

We now return to our original problem of "moving" b^j in front of a^i. Consider the string $v = xd^la^iy'y''z$. If $d^l = b^j$, then we are done. So, assume otherwise. Rewrite y'' as $y'' = y'''b^j$. Thus, we have $u = xa^iy'd^ly'''b^jz \in L(\mathcal{B})$ and $v = xd^la^iy'y'''b^jz \in L(\mathcal{B})$. As $|y'y'''| = N$ and $a, b \notin y'y'''$, we can apply the induction hypothesis to v. There are two cases:

1. There exist $y_1', y_1''' \in \mathcal{Y}^*$ such that $xd^ly_1'b^ja^iy_1'''z \in L(\mathcal{B})$ and $y_1'y_1''' \sim y'y'''$:

   Clearly, $d^ly_1'y_1''' \sim y'd^ly''' \sim y$, and we are done.

2. $a^i \rightsquigarrow_v^* b^j$:

   Then there exist $c^k \in a^iy'$ and $e^m \in y'''b^j$ such that $a^i \rightsquigarrow_v^* c^k$, $c^k \rightsquigarrow_v e^m$, and $e^m \rightsquigarrow_v^* b^j$. Clearly, $e^m \rightsquigarrow_u^* b^j$. We now show that $c^k \rightsquigarrow_u e^m$.

   Rewrite a^iy' as $y_1c^ky_2$ and $y'''b^j$ as $y_3e^my_4$, where $0 \leq |y_1|, |y_2|, |y_3|, |y_4| \leq N$. Thus, we have $u = xy_1c^ky_2d^ly_3e^my_4z$ and $v = xd^ly_1c^ky_2y_3e^my_4z$. Since $c^k \rightsquigarrow_v e^m$, we have $(c, e) \in \mathcal{F}$ and $c, e \notin y_2y_3$. If $d = e$, then $c^k \rightsquigarrow_u d^l$, a contradiction. If $d = c$, then $c^k \rightsquigarrow_u^* d^l$, by Lemma 4.3.1, a contradiction again. Therefore, $c, e \notin y_2d^ly_3$, and thus $c^k \rightsquigarrow_u e^m$. But then $c^k \rightsquigarrow_u^* b^j$, which contradicts with our choice of d^l. This completes our proof of the theorem.□

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 $a^i \rightsquigarrow_u^* a^j$.

4.3.2 The JTU-Rules

The first formal definition of delay-insensitivity was made by J. T. Udding in 1984 [19]. He used trace theory as the formalism for circuit specifications. An

alphabet is a finite set of symbols. Finite-length sequences of symbols in 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 module communicating with its environment. The input and output alphabets are two disjoint finite sets corresponding to the input and output terminals of the module. 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 module 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 module if that module is not ready to receive it. Similarly, the module 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 module 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 module, 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 [19], then there is no computation or transmission interference. These four rules along with prefix-closure property of trace set constitute the class of delay-insensitive trace structures.

4.3.3 Semi-Modularity Implies the JTU-Rules

We now investigate what semi-modularity of network behaviors imposes on the behaviors of individual modules. Let M be a module of a delay-dense network N, and suppose that M 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 considered 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. It is worth noting that this characterization does not apply to DELAY modules, since a module connected to a DELAY module does not necessarily function as an input or output terminal of that DELAY module, and transitions occurring on such a module may not reflect the input/output action of the DELAY module accurately.

Definition 4.3.3 Let N be a delay-dense network, and let M be a module of N which is not a DELAY module. Let the state variable corresponding to M be g.

- The *input alphabet* of M: $\mathbf{i}M = \{a \in \mathcal{Y} \mid (a, g) \in \mathcal{F}\}$;
- The *output alphabet* of M: $\mathbf{o}M = \{b \in \mathcal{Y} \mid (g, b) \in \mathcal{F}\}$;
- The *alphabet* of M: $\mathbf{a}M = \mathbf{i}M \cup \mathbf{o}M$.

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.

Definition 4.3.4 Let M and N be as defined in Definition 4.3.3. 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 $tr_M(B) = \langle \mathbf{i}M,\ \mathbf{o}M,\ L(\mathcal{B}) \downarrow \mathbf{a}M \rangle$, where $L(\mathcal{B}) \downarrow \mathbf{a}M = \{w \mid w = w' \downarrow \mathbf{a}M \text{ for some } w' \in L(\mathcal{B})\}$.

Theorem 4.4 *Let M and N be as defined in Definition 4.3.3. Suppose that $\mathbf{i}M \cap \mathbf{o}M = \emptyset$. 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 JTU-Rules hold for M, i.e.,*

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

Proof:

1. This is because $L(\mathcal{B})$ is prefix-closed and prefix-closure is preserved under projections (see Property 1.2 of [19]).
2. Assume that $uaa \in \mathbf{t}M$. Then there exist $u', x \in \mathcal{Y}^*$ such that $w = u'a^i x a^{i+1} \in L(\mathcal{B})$ 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, $uaa \notin \mathbf{t}M$.

3. 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 $w = u'a^i x b^j v' \in L(\mathcal{B})$, 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 $w' = u'x'b^j a^i x''v' \in L(\mathcal{B})$ 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, $uabv \in \mathbf{t}M \Longleftrightarrow ubav \in \mathbf{t}M$.

4. Let g be the state variable associated with the module M. We first prove an important lemma. For simplicity, we adopt the following notation

$$s \xrightarrow{y[u]} t$$

   for $s, t \in \mathcal{Q}$, $u \in \mathbf{t}M$, and $y \in \mathcal{Y}^*$ to denote that $t \in sy$ and $y \downarrow \mathbf{a}M = u$. Also, for two states $s, t \in \mathcal{Q}$, and symbol $a \in \mathcal{Y}$, we write $s =_a t$ if and only if $s_c = t_c$ for all $c \in \mathcal{Y}$ such that $c \neq a$.

   Lemma 4.4.1 *Let $p_1, p_3, r, t \in \mathcal{Q}$, $x, z \in \mathcal{Y}^*$, and $u \in \mathbf{t}M$ be such that*

$$p_1 \xrightarrow{x[u]} r \quad \textit{and} \quad p_3 \xrightarrow{z[u]} t.$$

   and $p_{1,c} = p_{3,c}$ for all $c \in \mathbf{a}M$. Let state p_2 be such that $p_{2,g} = p_{3,g}$ and $p_2 =_g p_1$. Note that then $p_{1,c} = p_{2,c} = p_{3,c}$ for all $c \in \mathbf{a}M$. If $p_2 \in \mathcal{Q}$, then there exists $s \in \mathcal{Q}$ such that

$$p_2 \xrightarrow{y[u]} s$$

for some $y \in \mathcal{Y}^*$, $s_g = t_g$, *and* $s =_g r$.

Proof: Intuitively, we construct the sequence y of state transitions to "imitate" z in regards to changes on g and to "imitate" x in regards to changes on all the other state variables.

We prove the lemma formally by induction on $|u|$. For the base case, let $u = \epsilon$. Then $x \downarrow \mathbf{a}M = \epsilon$. By the "Cycle" Corollary, g may occur in x at most once. Also, if $g \in x$, i.e., $x = x_1 g x_2$ for some x_1 and x_2, then by the corollary of Proposition 4.3.2, $r \in p_1 g x_1 x_2$. Let x be such that either $g \notin x$ or g is the first symbol of x. We make the same assumption about z. Define r' so that $r' = p_1$ if $g \notin x$ and if $g \in x$, then r' denotes the state reached after g in x. Also, define x' so that $x' = x$ if $g \notin x$ and if $g \in x$, then $x = gx'$. Define t' and z' to correspond to z similarly. Thus, we have either

$$p_1/r' \xrightarrow{x'[\epsilon]} r \quad \text{or} \quad p_1 \xrightarrow{g[\epsilon]} r' \xrightarrow{x'[\epsilon]} r,$$

and either

$$p_3/t' \xrightarrow{z'[\epsilon]} t \quad \text{or} \quad p_3 \xrightarrow{g[\epsilon]} t' \xrightarrow{z'[\epsilon]} t.$$

Note that $p_{2,g} = p_{3,g}$ and $p_{2,c} = p_{3,c}$ for all $c \in \mathbf{i}M$. It follows that $P_{2,g} = P_{3,g}$. Thus, $g \in \mathcal{U}(p_2)$ if and only if $g \in \mathcal{U}(p_3)$. Define s' so that $s' = p_2$ if $t' = p_3$ and otherwise, $s' \in p_2 g$ such that $s'_g = t'_g$. In either case, we have $s'_g = t'_g$. Clearly, $s' =_g p_2 =_g p_1 =_g r'$. Therefore, there exists $s \in s'x'$ such that $s =_g r$. Here, we imitate state transitions made in x' in exact order, starting in state s' instead of r'. This is allowed by the behavior since none of these transitions depend on the value of g, as $g \notin x'$ and $x' \downarrow \mathbf{o}M = \epsilon$. On the other hand, since $g \notin x'$, $s_g = s'_g = t'_g = t_g$. Define y so that $y = x'$ if $s' = p_2$ and otherwise, $y = gx'$. Then we have $s \in p_2 y$, $y \downarrow \mathbf{a}M = \epsilon$, $s_g = t_g$, and $s =_g r$, and we are done with the base case.

Now assume that the result holds for $|u| \leq k$. Let $u = u_1 d$, where $d \in \mathbf{a}M$ and $|u_1| = k$. We have

$$p_1 \xrightarrow{x_1[u_1]} r_1 \xrightarrow{d[d]} r_2 \xrightarrow{x_2[\epsilon]} r$$

and

$$p_3 \xrightarrow{z_1[u_1]} t_1 \xrightarrow{d[d]} t_2 \xrightarrow{z_2[\epsilon]} t.$$

By induction hypothesis, we have

$$p_2 \xrightarrow{y_1[u_1]} s_1$$

where $s_{1,g} = t_{1,g}$ and $s_1 =_g r_1$. Also, since $p_{1,c} = p_{2,c} = p_{3,c}$ for all $c \in \mathbf{a}M$ and all variables in $\mathbf{a}M$ are binary, we have $r_{1,c} = s_{1,c} = t_{1,c}$ for all $c \in \mathbf{a}M$. If $d \in \mathbf{i}M$, let $(c, d) \in \mathcal{F}$. Clearly, $c \neq g$ and $s_{1,c} = r_{1,c}$. It follows that $S_{1,d} = R_{1,d}$. Also, $s_{1,d} = r_{1,d}$ and $d \in \mathcal{U}(r_1)$. Thus, $d \in \mathcal{U}(s_1)$. If $d \in \mathbf{o}M$, then since $s_{1,g} = t_{1,g}$, we have $S_{1,d} = T_{1,d}$, and thus $d \in \mathcal{U}(s_1)$, again. Hence,

$$p_2 \xrightarrow{y_1[u_1]} s_1 \xrightarrow{d[d]} s_2.$$

Clearly, $s_{2,g} = t_{2,g}$, $s_2 =_g r_2$, and $r_{2,c} = s_{2,c} = t_{2,c}$ for all $c \in \mathbf{a}M$.

Following the proof of the base case, we conclude that there exists $s \in s_2 y_2$ for some $y_2 \in \mathcal{Y}^*$ such that $y_2 \downarrow \mathbf{a}M = \epsilon$, $s_g = t_g$, and $s =_g r$. Hence, $s \in p_2 y$, where $y = y_1 d y_2$, such that $y \downarrow \mathbf{a}M = u$, $s_g = t_g$, and $s =_g r$. This completes the proof of the lemma. □

Now, let us prove 4. By definition, we have

$$ua \in \mathbf{t}M \implies q \xrightarrow{x[u]} r \xrightarrow{a[a]} r'$$

and

$$ub \in \mathbf{t}M \implies q \xrightarrow{z[u]} t \xrightarrow{b[b]} t'.$$

Without loss of generality, assume that $a \in \mathbf{i}M$ and $b \in \mathbf{o}M$. By Lemma 4.4.1, we have

$$q \xrightarrow{y[u]} s$$

such that $s_g = t_g$ and $s =_g r$. Also, $r_c = s_c = t_c$ for all $c \in \mathbf{a}M$. It is not hard to see that $a, b \in \mathcal{U}(s)$. By semi-modularity, $sab \neq \emptyset$ and $sba \neq \emptyset$, and thus $yab \in L(\mathcal{B})$ and $yba \in L(\mathcal{B})$, which further implies that $uab \in \mathbf{t}M$ and $uba \in \mathbf{t}M$. We are done.

5. Assume that $uabvc \in \mathbf{t}M$ and $ubav \in \mathbf{t}M$ for traces u and v, and symbols $a, b, c \in \mathbf{a}M$, where b is of another type than a and c. Again, let g be the state variable associated with module M. By definition, we have

$$q \xrightarrow{x_1[u]} r_1 \xrightarrow{a[a]} r_2 \xrightarrow{x_2[\epsilon]} r_3 \xrightarrow{b[b]} r_4 \xrightarrow{x_3[v]} r_5 \xrightarrow{c[c]} r_6$$

and

$$q \xrightarrow{z_1[u]} t_1 \xrightarrow{b[b]} t_2 \xrightarrow{z_2[\epsilon]} t_3 \xrightarrow{a[a]} t_4 \xrightarrow{z_3[v]} t_5.$$

Let $x = x_1 a^i x_2 b^j x_3 c^k$ and $z = z_1 b^j z_2 a^i z_3$. Then, $x, z \in L(\mathcal{B})$. There are two cases to consider.

- $b \in \mathbf{o}M$ and $a, c \in \mathbf{i}M$:

  Then $(a, g), (c, g) \in \mathcal{F}$ and $(g, b) \in \mathcal{F}$. If $g \notin x_2$, then $a^i \not\rightarrow_x^* b^j$. By the "Chain" Theorem, there exist $x_2', x_2'' \in \mathcal{Y}^*$ such that $x_1 x_2' b a x_2'' x_3 c \in L(\mathcal{B})$ and $x_2' x_2'' \sim x_2$, which implies that $ubavc \in \mathbf{t}M$. If $g \in x_2$, by the corollary of Proposition 4.3.2, we can always move g to the front. That is, $x_1 a^i g x_2' b^j x_3 c^k \in L(\mathcal{B})$ and $g x_2' \sim x_2$. Clearly, $g \notin x_2'$. By the same corollary again, we have

$$q \xrightarrow{x_1[u]} r_1 \xrightarrow{agb[ab]} r_1' \xrightarrow{x_2' x_3[v]} r_5 \xrightarrow{c[c]} r_6.$$

Similarly, if $g \in z_2$, then we can move that g in front of b^j (recall that $b \in \mathbf{o}M$). Therefore, we simply assume that $g \notin z_2$.

By Lemma 4.4.1, we have

$$q \xrightarrow{y_1[u]} s_1$$

where $s_{1,g} = t_{1,g}$ and $s_1 =_g r_1$. Also, $r_{1,d} = s_{1,d} = t_{1,d}$ for all $d \in \mathbf{a}M$. Clearly, $a, b \in \mathcal{U}(s_1)$. Now, changing b in state s_1, and then changing a, we obtain

$$q \xrightarrow{y_1[u]} s_1 \xrightarrow{ba[ba]} s_2.$$

Clearly, $s_2 =_g r'_1$. On the other hand, $s_{2,g} = s_{1,g} = t_{1,g} = t_{4,g}$, since $g \notin z_2$. Also, $r'_{1,d} = s_{2,d} = t_{4,d}$ for all $d \in \mathbf{a}M$. By Lemma 4.4.1, there exists $s_3 \in s_2 y_2$, where $y_2 \in \mathcal{Y}^*$ and $y_2 \downarrow \mathbf{a}M = v$, such that $s_{3,g} = t_{5,g}$ and $s_3 =_g r_5$. Clearly, $c \in \mathcal{U}(s_3)$. Hence,

$$q \xrightarrow{y_1[u]} s_1 \xrightarrow{ba[ba]} s_2 \xrightarrow{y_2[v]} s_3 \xrightarrow{c[c]} s_4$$

and $ubavc \in \mathbf{t}M$, as required.

- $b \in \mathbf{i}M$ and $a, c \in \mathbf{o}M$:

  Then $(b, g) \in \mathcal{F}$ and $(g, a), (g, c) \in \mathcal{F}$. We only need to prove the result under the following assumptions:

  - $g \notin x_2$;
  - if $g \in z_2$ (g is unique), then g is the first symbol of z_2.

  Define t'_2 so that $t'_2 = t_2$ if $g \notin z_2$ and if $g \in z_2$, then t'_2 denotes the state reached after g in z_2. Also, define z'_2 so that $z'_2 = z_2$ if $g \notin z_2$ and otherwise, $z_2 = g z'_2$. Thus, $g \notin z'_2$. By the corollary of Proposition 4.3.2, we can move a^i in front of z'_2. Therefore, we have either

$$q \xrightarrow{z_1[u]} t_1 \xrightarrow{b[b]} t_2/t'_2 \xrightarrow{a[a]} t'_3 \xrightarrow{z'_2 z_3[v]} t_5$$

or

$$q \xrightarrow{z_1[u]} t_1 \xrightarrow{b[b]} t_2 \xrightarrow{g[\epsilon]} t'_2 \xrightarrow{a[a]} t'_3 \xrightarrow{z'_2 z_3[v]} t_5.$$

By Lemma 4.4.1, we have

$$q \xrightarrow{y_1[u]} s_1$$

such that $s_{1,g} = r_{1,g}$ and $s_1 =_g t_1$. Also, $r_{1,d} = s_{1,d} = t_{1,d}$ for all $d \in \mathbf{a}M$. Clearly, $a, b \in \mathcal{U}(s_1)$. Now, in state s_1, changing b and then a, we obtain

$$q \xrightarrow{y_1[u]} s_1 \xrightarrow{ba[ba]} s_2.$$

Clearly, $s_2 =_g t'_3$. On the other hand, $s_{2,g} = s_{1,g} = r_{1,g} = r_{4,g}$, since $g \notin x_2$. Also, $r_{4,d} = s_{2,d} = t'_{3,d}$ for all $d \in \mathbf{a}M$. By Lemma 4.4.1, there exists $s_3 \in s_2 y_2$, where $y_2 \in \mathcal{Y}^*$ and $y_2 \downarrow \mathbf{a}M = v$, such that $s_{3,g} = r_{5,g}$ and $s_3 =_g t_5$. Clearly, $c \in \mathcal{U}(s_3)$. Hence,

$$q \xrightarrow{y_1[u]} s_1 \xrightarrow{ba[ba]} s_2 \xrightarrow{y_2[v]} s_3 \xrightarrow{c[c]} s_4$$

and $ubavc \in \mathbf{t}M$, as required. We are done. □

Theorem 4.4 states that if a behavior of a delay-dense network is semi-modular, then the trace structure of each "non-DELAY" module induced by the behavior must be delay-insensitive, in the sense that it satisfies the JTU-Rules. It is worth noting that the converse of this theorem is not true, as the following example demonstrates.

Consider the network N_4 shown in Figure 4.5. Modules a, b, c, and d are DELAY modules. The functions of modules M_1 and M_2 are shown in Tables 4.1 and 4.2, respectively. We denote the internal state variables of M_1 and M_2 by g_1 and g_2, respectively.

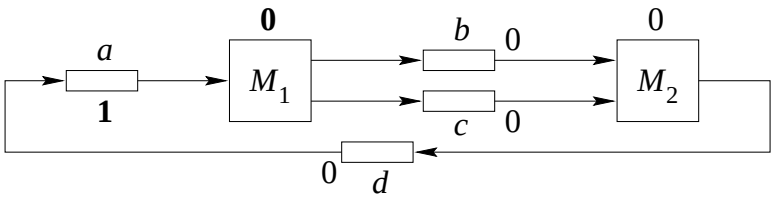


Figure 4.5: Network N_4.

Table 4.1: Functions of M_1.

	a		
g_1	0	1	bc
0	2	1	00
1	1	1	10
2	2	2	11

$\delta(a, g_1)$

Table 4.2: Functions of M_2.

	bc				
g_2	00	01	10	11	d
0	0	0	0	0	0

$\delta(bc, g_2)$

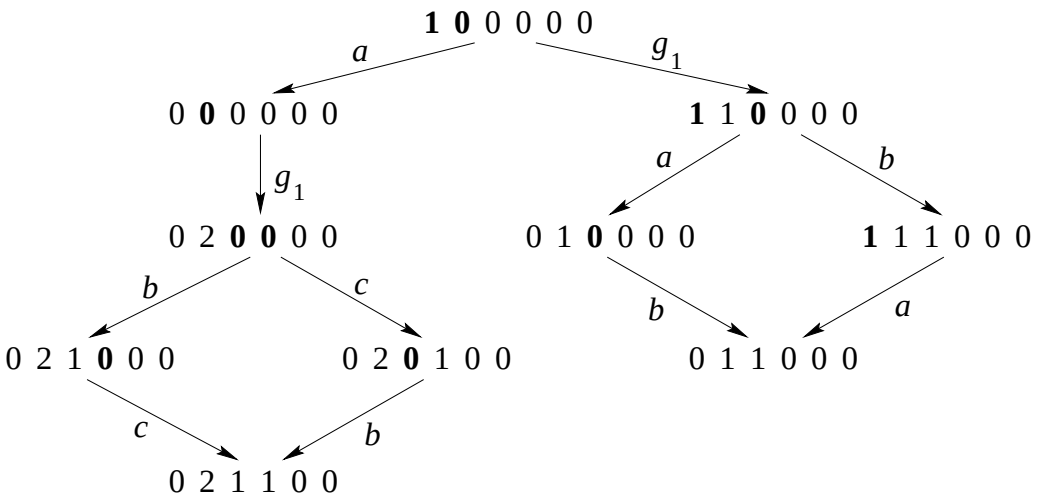


Figure 4.6: A behavior of network N_4.

Let the initial state be $ag_1bcg_2d = \mathbf{1}00000$. The corresponding behavior is shown in Figure 4.6. The trace structure of M_1 induced by the behavior is

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

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; more specifically, the initial state is not semi-modular.

Chapter 5

Delay-Insensitivity and Semi-Modularity

In this chapter, we prove several results relating network delay-insensitivity to semi-modularity. It turns out that these two properties are equivalent for delay-dense networks. That is, 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. We also discuss some issues involved in testing whether a given network is delay-insensitive.

5.1 Delay-Insensitivity and Semi-Modularity

We first prove that delay-insensitivity implies semi-modularity; here, delay-density is not required. We then prove that for delay-dense networks, delay-insensitivity is equivalent to semi-modularity.

Theorem 5.1 *If a network is delay-insensitive with respect to a state, then its behavior originating from that state is semi-modular.*

Proof: Let N be a network, which is delay-insensitive with respect to a state q. Let $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ be the corresponding behavior. Let $\hat{N} \in \mathcal{D}(N)$ be obtained from N by inserting one DELAY module in *each* connection of N. Let $\hat{B} = \langle \hat{q}, \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ be the behavior of $\hat{N}$ which is initial-state compatible with B. By Theorem 3.1, $\hat{B}$ is delay-safe with respect to B. We first prove the following claim.

Lemma 5.1.1 *Let $s \in \mathcal{Q}$. There exists $\hat{s} \in \hat{\mathcal{Q}}$ such that $\hat{s}$ is the stable-delay extension of s.*

Proof: We prove the result by induction. The base case, where $s = q$, is trivial. Let $s \in \mathcal{Q}$ and $t \in sy^i$, for some $y^i \in \mathcal{Y}$. Assume that there exists $\hat{s} \in \hat{\mathcal{Q}}$ such that $\hat{s}$ is the stable-delay extension of s. By Proposition 3.1.1, $\hat{S}_i = S_i$. Thus, there exists $\hat{t} \in \hat{\mathcal{Q}}$ such that $\hat{t} \in \hat{s}y^i$ and $\hat{t} \downarrow \mathcal{Y} = t$. By Proposition 3.1.2, there exists $w \in (\hat{\mathcal{Y}} - \mathcal{Y})^*$ such that $\hat{t}w = \{\hat{t}'\}$, where $\hat{t}' \in \hat{\mathcal{Q}}$ is the stable-delay extension of t. Thus, the lemma is proved. □

Now suppose that B is not semi-modular. Then there exists $(r, s) \in \mathcal{R}$ with $\tau(r, s) = y^i$ such that $R_j \not\subseteq S_j$ for some $j \neq i$ with $y^j \in \mathcal{U}(r)$. Let $b \in R_j$ such that $b \notin S_j$. By Lemma 5.1.1, there exists $\hat{r} \in \hat{\mathcal{Q}}$ such that $\hat{r}$ is the stable-delay extension of r. By Proposition 3.1.1, $R_i = \hat{R}_i$ and $R_j = \hat{R}_j$. Thus, there exists $\hat{s} \in \hat{\mathcal{Q}}$ such that $\hat{s} \in \hat{r}y^i$ and $\hat{s} \downarrow \mathcal{Y} = s$. By our construction of $\hat{N}$, $(y^i, y^j) \notin \hat{\mathcal{F}}$. By Proposition 2.3.1, we have $\hat{R}_j = \hat{S}_j$. Note that $y^j \in \mathcal{U}(r) = \mathcal{U}(\hat{r})$. It follows that $y^j \in \mathcal{U}(\hat{s})$. Also, $b \in R_j = \hat{R}_j = \hat{S}_j$. Thus, there exists $\hat{t} \in \hat{\mathcal{Q}}$ such that $\hat{t} \in \hat{s}y^j$ and $\hat{t}_j = b$. Recall that $\hat{s}$ is an extension of s and $\hat{B}$ is delay-safe with respect to B. Therefore, there exists $t \in \mathcal{Q}$ such that $t \in sy^j$ and $t = \hat{t} \downarrow \mathcal{Y}$. Thus, $t_j = b$, which implies that $b \in S_j$—a contradiction. Hence, B must be semi-modular. □

The next theorem asserts that if the behavior of a delay extension is semi-modular, then it is delay-safe with respect to the behavior of the original network.

Theorem 5.2 *Let $\hat{N} \in \mathcal{D}(N)$. Let $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ be a behavior of N, and let $\hat{B} = \langle \hat{q}, \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ be the behavior of $\hat{N}$ which is initial-state compatible with B. If $\hat{B}$ is semi-modular, then it is delay-safe with respect to B.*

Proof: Let $\hat{s} \in \hat{\mathcal{Q}}$ be an extension of $s \in \mathcal{Q}$. Let $\hat{t} \in \hat{\mathcal{Q}}$ and $\hat{w} \in \hat{\mathcal{Y}}^*$ be such that $\hat{t} \in \hat{s}\hat{w}$ and $\hat{w} \downarrow \mathcal{Y} = y^i$ for some $y^i \in \mathcal{Y}$. Let $\hat{t}_i = b$. Clearly, all we need to show is that $b \in S_i$. Let us write $\hat{w}$ as $\hat{w} = \hat{u}y^i\hat{u}'$, where $\hat{u}, \hat{u}' \in (\hat{\mathcal{Y}} - \mathcal{Y})^*$. Denote the state reached after $\hat{u}$ by $\hat{s}'$, $\hat{s}' \in \hat{s}\hat{u}$. Clearly, $y^i \in \mathcal{U}(\hat{s}')$ and $b \in \hat{S}'_i$. Also, $\hat{s}'$ is an extension of s. By Proposition 3.1.2, there exists $\hat{v} \in (\hat{\mathcal{Y}} - \mathcal{Y})^*$ such that $\hat{s}'\hat{v} = \{\hat{s}''\}$, where $\hat{s}''$ is the stable-delay extension of s. Note that $\hat{B}$ is semi-modular. By Proposition 4.1.1, we have $\hat{S}'_i \subseteq \hat{S}''_i$, which implies that $b \in \hat{S}''_i$. On the other hand, by Proposition 3.1.1, we have $S_i = \hat{S}''_i$. Hence, $b \in S_i$. By definition, $\hat{B}$ is delay-safe with respect to B. □

Theorem 5.3 *If a behavior of a delay-dense network is semi-modular, then the network is delay-insensitive with respect to the initial state of that behavior.*

Proof: Let N be a delay-dense network. Let $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ be a semi-modular behavior of N. Let $\hat{N} \in \mathcal{D}(N)$, and let $\hat{B}$ be the behavior of $\hat{N}$ which is initial-state compatible with B. By Theorem 4.2, $\hat{B}$ is semi-modular. By Theorem 5.2, $\hat{B}$ is delay-safe with respect to B. Since $\hat{N}$ was chosen arbitrarily, by Theorem 3.1, N is delay-insensitive with respect to q. □

Combining Theorem 5.1 and 5.3, we conclude that delay-insensitivity and semi-modularity are equivalent for delay-dense networks.

Theorem 5.4 *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.*

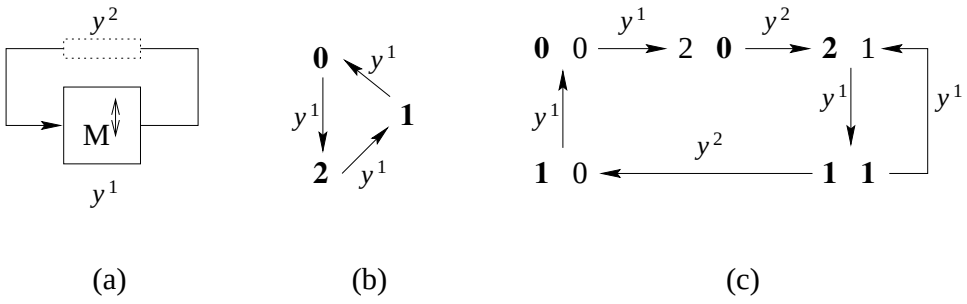


Figure 5.1: Theorem 5.3 fails to hold for networks which are not delay-dense.

Note that Theorem 5.3 does not necessarily hold for networks which are not delay-dense. For example, consider the network N consisting of a single module $M^{\updownarrow}$ (as defined in Table 2.9), as shown in Figure 5.1(a). Let the initial state be **0**. The corresponding behavior B is semi-modular, as shown in Figure 5.1(b). Now let us insert a DELAY module y^2, as shown in Figure 5.1(a). The corresponding behavior $\hat{B}$ of the resulting delay extension $\hat{N}$ is given in Figure 5.1(c). Note that in state **11** of $\hat{B}$, $M^{\updownarrow}$ may change to 2; but in state **1** of B, $M^{\updownarrow}$ can only change to 0. Therefore, $\hat{B}$ is not delay-safe with respect to B, and consequently, N is not delay-insensitive with respect to state **0**.

It is also worth noting that delay-insensitivity is not preserved under delay extensions in general. For example, consider the network of Figure 4.3(a) (without the DELAY module y^3). It is not hard to verify that it is delay-insensitive with respect to the state **0**0, because the AND-gate never changes its state and the INVERTER module oscillates in any delay extension. However, with the DELAY module y^3 inserted, the resulting behavior (as shown in Figure 4.3(c)) is not even semi-modular. Hence, the resulting network is not delay-insensitive, by Theorem 5.1.

5.2 Testing Delay-Insensitivity of Networks

Suppose that we are given a network N and an initial state q, and we are to determine whether N is delay-insensitive with respect to q. We cannot apply the definition of network delay-insensitivity directly, since it would involve an infinite test. Testing the specification of each network module against the JTU-Rules would also involve a potentially infinite test, since the trace sets are usually infinite. If N happens to be delay-dense, then by Theorem 5.4, we only need to check whether the behavior of N originating from q is semi-modular; this problem is decidable. The test for semi-modularity is linear in the size of the behavior graph, but exponential in the number of state variables. Here, we assume that the size of the domain of each state variable is constant.

We do not have a complete test for all networks. However, we do have one positive test and one negative test; both involve testing for semi-modularity only. The negative test uses Theorem 5.1: If the behavior of N originating from q is not semi-modular, then N is not delay-insensitive with respect to q. The positive test is concerned with *delay-completion*.

Definition 5.2.1 The *delay-completion* of a network N is a delay extension of N obtained by inserting one DELAY module in each connection of N.

Theorem 5.5 *Let N' be the delay-completion of a network N. Let $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ be a behavior of N, and let $B' = \langle q', \mathcal{Q}', \mathcal{R}' \rangle$ be the behavior of N' which is initial-state compatible with B. If B' is semi-modular, then N is delay-insensitive with respect to q.*

Proof: Let $\hat{N} \in \mathcal{D}(N)$, and let $\hat{B} = \langle \hat{q}, \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ be the behavior of $\hat{N}$ which is initial-state compatible with B. Clearly, there exists a network $\hat{N}'$ which is a delay

extension of both $\hat{N}$ and N'. For example, $\hat{N}'$ can be the delay-completion of $\hat{N}$. Let $\hat{B}' = \langle \hat{q'}, \hat{\mathcal{Q}}', \hat{\mathcal{R}}' \rangle$ be the behavior of $\hat{N}'$ which is initial-state compatible with B. One easily verifies that $\hat{B}'$ is initial-state compatible with both B' and $\hat{B}$.

We shall prove that $\hat{B}$ is delay-safe with respect to B if B' is semi-modular. Let $\hat{s} \in \hat{\mathcal{Q}}$ be an extension of $s \in \mathcal{Q}$. Let $\hat{t} \in \hat{\mathcal{Q}}$ and $\hat{w} \in \hat{\mathcal{Y}}^*$ be such that $\hat{t} \in \hat{s}\hat{w}$ and $\hat{w} \downarrow \mathcal{Y} = y^i$ for some $y^i \in \mathcal{Y}$. Let $\hat{t}_i = b$. Clearly, all we need to show is that $b \in S_i$. Let us write $\hat{w}$ as $\hat{w} = \hat{u_1} y^i \hat{u_2}$, where $\hat{u_1}, \hat{u_2} \in (\hat{\mathcal{Y}} - \mathcal{Y})^*$. Denote the state reached after $\hat{u_1}$ by $\hat{r}$, $\hat{r} \in \hat{s}\hat{u_1}$. Clearly, $b \in \hat{R}_i$.

By Lemma 5.1.1, there exists $\hat{s'} \in \hat{\mathcal{Q}}'$ such that $\hat{s'}$ is the stable-delay extension of $\hat{r}$. We have $b \in \hat{S'}_i = \hat{R}_i$. Note that N' is delay-dense and B' is semi-modular. By Theorem 4.2, $\hat{B}'$ is semi-modular. It is not hard to see that $\hat{s'} \downarrow \mathcal{Y}' \in \mathcal{Q}'$. Let $s' = \hat{s'} \downarrow \mathcal{Y}'$. By Theorem 5.2, $\hat{B}'$ is delay-safe with respect to B'. Thus, $b \in S'_i$. By Theorem 5.2 again, B' is delay-safe with respect to B. Also, $s' \downarrow \mathcal{Y} = s$. Therefore, $b \in S_i$. We are done. □

Thus, given N and q, we first construct the delay-completion of N. We then check whether the behavior of the delay-completion is semi-modular. If the answer is yes, then by Theorem 5.5, we conclude that N is delay-insensitive with respect to q. Unfortunately, there are networks for which neither the positive test nor the negative test gives a definitive answer. An example of such a network was given in Figure 4.3(a).

So far, we have assumed that the network model is already given. Now suppose that we are simply given an autonomous asynchronous circuit whose components can be modelled by modules. We need to select a network model for the circuit first. As we pointed out in Section 4.2.1, we have to take into account both component delays and wire delays. Therefore, the circuit corresponds to a delay-dense network,

for which testing for delay-insensitivity is equivalent to testing for semi-modularity. One may argue against this, since violation of delay-safety or semi-modularity may occur only on an added DELAY module, and this violation may not affect the operations of the circuit components (e.g., the network in Figure 4.3). But violation of semi-modularity on a DELAY module corresponds to a situation where two consecutive binary signals are sent along the same wire (transmission interference), which is generally considered a violation of delay-insensitivity.

Chapter 6

Concluding Remarks

We conclude the thesis with a few open problems and a classification of autonomous asynchronous networks using the criteria we discussed, i.e., delay-insensitivity, semi-modularity, and delay-density.

6.1 Open Problems

1. Is there an algorithm which determines whether a given network that is not delay-dense is delay-insensitive?

   It is still desirable to find such a test, since one may choose to tolerate transmission interference in a circuit. We conjecture that a network is delay-insensitive if and only if the behavior of the delay-completion of the network is delay-safe.

2. Can our definition of semi-modularity be related to some lattice-theoretic properties, as in the case of deterministic behaviors [11]?

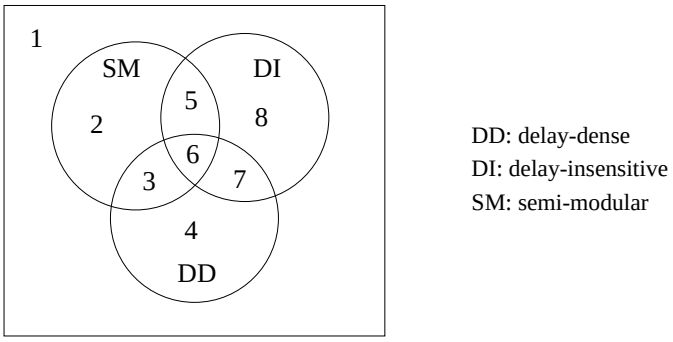


Figure 6.1: Classification of asynchronous networks.

3. Are there efficient algorithms for testing whether a given behavior is semi-modular?

6.2 Classification of Asynchronous Networks

Figure 6.1 shows a classification of autonomous asynchronous networks. The universal set is really the set of autonomous networks with designated initial states.

We now describe each numbered region in Figure 6.1:

1. nonempty: a network belonging to this region is shown in Figure 6.2(a);
2. nonempty: an example was given in Figure 5.1(a) (without the DELAY module y^2);
3. empty: by Theorem 5.3;
4. nonempty: an example is shown in Figure 6.2(b);

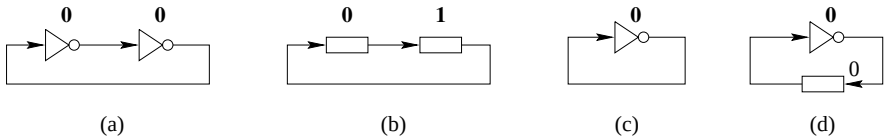


Figure 6.2: Network examples for Figure 6.1.

5. nonempty: an example is shown in Figure 6.2(c);

6. nonempty: an example is shown in Figure 6.2(d);

7. empty: by Theorem 5.1;

8. empty: by Theorem 5.1.

In our classification, we can also include speed-independent circuits, as defined by Muller [11]. According to Muller, a circuit is speed-independent with respect to a state s if all *allowed state sequences* starting with s have the same *terminal class.* An allowed state sequence corresponds to a path in the behavior of the circuit; it is finite only if the last state of the sequence is stable; it cannot end in a *transient* cycle. A cycle of states is said to be transient if there is a state variable which is unstable and remains unchanged throughout the cycle. Let $\mathcal{R}^*$ be the reflexive and transitive closure of $\mathcal{R}$ (Definition 2.3.4). Two states a and b are equivalent if $a\mathcal{R}^*b$ and $b\mathcal{R}^*a$. Let $\mathcal{L}$ be a partial order defined over the induced equivalence classes: $A\mathcal{L}B$ if there exists $a \in A$ and $b \in B$ such that $a\mathcal{R}^*b$. For each allowed state sequence, there is a unique sequence of equivalence classes and a definite "last" class, the terminal class.

Muller considered only deterministic modules. In our network model, we call a network deterministic if all the modules in the network are deterministic; otherwise,

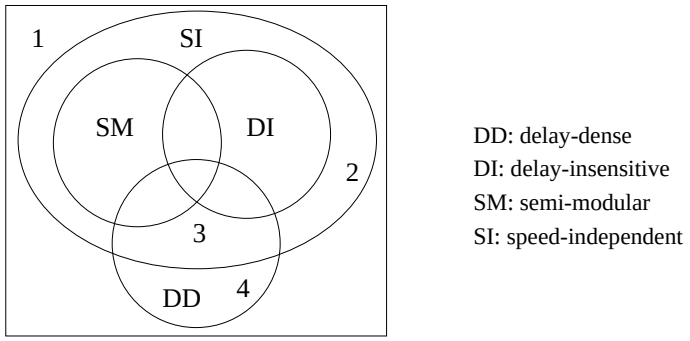


Figure 6.3: Classification of deterministic asynchronous networks.

it is nondeterministic. Muller showed that circuits exhibiting semi-modular behaviors form a *proper* subclass of speed-independent circuits. Using Muller's definition, we see that the circuit of Figure 4.2 is certainly not speed-independent; however, according to our definition, the circuit is semi-modular. This does not contradict Muller's result, for our definition of semi-modularity is less restrictive than that of Muller's. For deterministic networks, however, the two definitions are equivalent. Figure 6.3 shows the classification of deterministic networks.

We now describe each numbered region in Figure 6.3:

1. nonempty: a network belonging to this region was given in Figure 6.2(a);
2. nonempty: an example is shown in Figure 6.4(a) (from [11]);
3. nonempty: an example is shown in Figure 6.4(b);
4. nonempty: an example was given in Figure 6.2(b).

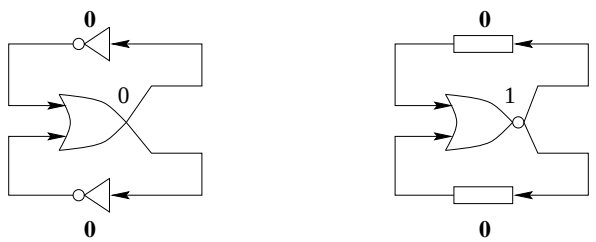


Figure 6.4: Network examples for Figure 6.3.

References

[1] J. A. Brzozowski, "Delay-Insensitivity and Ternary Simulation," *Proceedings of the First International Conference on Semigroups & Algebraic Engineering,*, Aizu-Wakamatsu City, Japan, March 24–38, 1997.

[2] J. A. Brzozowski and J. C. Ebergen, "On the Delay-Sensitivity of Gate Networks," *IEEE Trans. on Computers,* vol. 41, no. 11, pp. 1349–1360, November 1992.

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

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

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

[6] 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.

[7] J. C. Ebergen, "A Formal Approach to Designing Delay-Insensitive Circuits," *Distributed Computing*, vol. 5, no. 3, pp. 107–119, 1991.

[8] A. J. Martin. Programming in VLSI: From Communicating Processes to Delay-Insensitive Circuits. In C. A. R. Hoare, editor, *UT Year of Programming Institute on Concurrent Programming.* Addison-Wesley Publishing Company, 1989.

[9] A. J. Martin, "The Limitations to Delay-Insensitivity in Asynchronous Circuits," *Proceedings of the 6th MIT Conference on Advanced Research in VLSI,* W. J. Dally, ed., MIT Press, pp. 263–278, 1990.

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

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

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

[13] 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.

[14] E. F. Moore, "Gedanken Experiments on Sequential Machines," in C. E. Shannon and J. McCarthy, eds., *Automata Studies, Annals of Mathematics Study 34,* Princeton University Press, Princeton NJ, pp. 129–153, 1956.

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

[16] C-J. H. Seger, "On the Existence of Speed-Independent Circuits," *Theoretical Computer Science,* vol. 86, no. 2, pp. 343–364, 1991.

[17] C. L. Seitz, "System Timing," Chapter 7 in [10], pp. 218–262.

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

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

[20] 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.

[21] S. H. Unger, *Asynchronous Sequential Switching Circuits*, Wiley-Interscience, New York, 1969.

[22] 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.

[23] T. Verhoeff, *A Theory of Delay-Insensitive Systems*, PhD Thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands, May 1994.