Finite Buffer Realization of Input-Output Discrete Event Systems [1,2]

Ratnesh Kumar
Department of Electrical Engineering
University of Kentucky
Lexington, KY 40506-0046
Email: kumar@engr.uky.edu

Vijay K. Garg
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas 78712-1084
Email: vijay@pine.ece.utexas.edu

Steven I. Marcus
Department of Electrical Engineering and
Institute for Systems Research
University of Maryland
College Park, MD 20742
Email: marcus@src.umd.edu

[1]A preliminary version of the paper has appeared in [16].

[2]This research was supported in part by the Center for Robotics and Manufacturing, University of Kentucky, in part by the National Science Foundation under Grants NSF-ECS-9409712, NSFD-CDR-8803012, and NSF-CCR-9110605, in part by the Air Force Office of Scientific Research (AFOSR) under Contract F49620-92-J-0045, and in part by a TRW faculty assistantship award.

Abstract

Many discrete event systems (DESs) such as manufacturing systems, database management systems, communication networks, traffic systems, etc., can be modeled as input-output discrete event systems (I/O DESs). In this paper we formulate and study the problem of stable realization of such systems in the logical setting. Given an *input* and an *output* language describing the sequences of events that occur at the input and the output, respectively, of an I/O DES, we study whether it is possible to realize the system as a unit consisting of a given set of buffers of finite capacity, called a *dispatching unit*. The notions of stable, conditionally stable, dispatchable and conditionally dispatchable units are introduced as existence of stable (or input-output bounded), and causal (or prefix preserving) input-output maps, and effectively computable necessary and sufficient conditions for testing them are obtained.

1 Introduction and Motivation

A discrete event system (DES) is one in which the dynamics or evolution of the system is governed by the occurrences of certain discrete qualitative changes, called *events*, in the system. Arrival of a customer in a queueing system, loss of a message packet in a communication network, termination of an algorithm in a computer program, etc., are some examples of events. A logical framework was introduced by Ramadge and Wonham [26, 14] for studying the control of the qualitative behavior of DESs, where a DES was modeled as a state machine [10], and the logical behavior of the DES was described using the language or the set of all possible execution sequences of the state machine.

Many DESs such as manufacturing systems, communication networks, database management systems, traffic systems, etc., can be viewed as input-output discrete event systems (I/O DESs), where an "input" and an "output" language describe the logical behavior at the input and the output, respectively, of the DES. In a communication network, for example, the collection of all sequences in which messages arrive at the network defines the input language, and the collection of all sequences in which messages depart from the network defines the output language. In a database management system, the set of all interleavings of transaction operations constitutes the input language, whereas the set of all *serializable* and *strict* [23] interleavings of transaction operations constitutes the output language. Similarly, in a manufacturing system, the set of all possible parts arrival sequences corresponds to the logical behavior at the input, and the set of all *desired* parts departure sequences corresponds to the logical behavior at the output end. Common to all these systems is the fact that the events at the input/output end correspond to *physical* arrival/departure of messages/read-write operations/parts/customers, etc. In the computer science literature, models such as Moore and Mealy automata [10, Section 2.7], I/O automata [20], etc., and more recently in the DES control literature, models such as "command-response" systems [1] have been proposed for studying I/O DESs.

In this paper we study the issue of stable realization of a given I/O DES. Although the work presented here is applicable to any I/O DES of the type described above, our terminology pertains to manufacturing systems. (Consequently, the input and output languages consist of parts arrival and departure sequences, respectively.) Suppose n different types of parts $P_1, P_2, \ldots, P_n$ arrive in some order (on one or more conveyor belts) at the input of a manufacturing system, whereas they depart in some other order at the output. We investigate whether such an input-output manufacturing system can be represented as a dispatching unit consisting of n finite capacity buffers $B_1, B_2, \ldots, B_n$ (buffer B_k is used for storing part type P_k) together with a *dispatching policy*—a *stable* and *causal* input-output map. The dispatching policy determines at each arrival instance whether or not to buffer the arrived part, and the sequence in which buffered parts must be dispatched so that the buffer capacity constraints remain satisfied at each such instance. Note that our model allows the possibility of simultaneous arrival and sequence of departures. Also note that if the departure sequence at a certain arrival instance is the empty sequence, then this simply means that the arrived part is buffered in its buffer at that instance.

It is evident that if there exists no departure sequence constraint, or equivalently, if the output language consists of the collection of all possible departure sequences, then there exists a trivial dispatching policy, which dispatches a part when it arrives; thus no buffering is needed. Hence if there is no departure sequence constraint, then any given unit is "dispatchable". However, when a departure sequence constraint is imposed, then there is no a priori guarantee that a dispatching unit will be dispatchable. For example, suppose that two types of parts a and b arrive in the order $ababab\ldots$, whereas they must be dispatched in the order $aabaabaab\ldots$. Since part types a and b arrive alternately, whereas two parts of type a and one part of type b are dispatched alternately, it is clear that the buffer for part type b will grow without bound.

It follows from this example that the stability of the buffers in the dispatching unit depends on the *qualitative* constraint imposed by the set of arrival and departure sequences. This observation motivates the work presented in this paper. The work presented here thus supplements other works such as those reported in [25, 13, 19, 12, 3, 27, 6, 5], which deal with the *quantitative* stability issues for manufacturing systems. The works reported in [2, 22, 21, 15] also deal with stability issues in the logical setting. However, these address the issue of eventual/infinitely-often reachability of "safe" states/strings, and are not applicable to the study of boundedness of buffers as described above. It should also be noted that the dispatching unit that we study in this paper is an example of a vector discrete event system for which the *controllability* issues have recently been reported in [17, 18]. Our work deals with *stability* issues of such systems.

A dispatching unit is defined as a triple $(I, O, \overrightarrow{|B|})$, where I, called the *input language*, denotes the set of all possible arrival sequences; O, called the *output language*, denotes the set of all possible departure sequences; and $\overrightarrow{|B|}$, called the *buffer capacity vector*, represents the capacities of the individual buffers. We consider two kinds of scenarios: (i) The arrival sequence is known in advance. This applies, for example, to the setting of a manufacturing system where production planning is done in an off-line manner—i.e., prior to the actual production. This information should be taken into account for designing a dispatching policy. The notions of conditional stability and conditional dispatchability discussed below refer to this scenario. (ii) The arrival sequence is not known in advance, and the decision to either temporarily buffer an arrived part, or to dispatch it, is taken based on the previously arrived sequence of parts. This applies, for example, to the setting of a flexible manufacturing system where production decisions are made in an on-line manner—i.e., during the actual run of production. The notions of (unconditional) stability and dispatchability discussed below refer to this scenario.

Given a language $I' \subseteq I$, a map $\mathcal{D}_{I'} : I' \rightarrow O$ is said to be an *input-output map over* I'; if $I' = I$, then it is simply said to be an input-output map. An input-output map cannot be used to realize a dispatching policy unless it is causal (or prefix preserving) and stable (or input-output bounded). In case an input-output map is causal and stable, we provide a technique for realizing a dispatching policy. Causality of an input-output map is needed to unambiguously determine the sequence of parts to be dispatched at each arrival instance, whereas stability of an input-output map is needed to satisfy the buffer capacity constraints at each such instance. A dispatching unit is defined to be stable if there exists a stable input-output map for it. A dispatching unit is defined to be dispatchable if there exists a stable as well as causal input-output map for it.

In the setting where the arrival sequence, say $s \in I$, is known in advance, then an input-output map corresponds to a map of the type $\mathcal{D}_{\overline{s}} : \overline{s} \to O$ defined from the set of prefixes of the known arrival sequence to the output language. A dispatching policy in this setting is a causal and stable input-output map defined over the prefixes of the known arrival sequence. A dispatching unit is defined to be conditionally stable if for each arrival sequence there exists a stable input-output map defined over the prefixes of the arrival sequence. A dispatching unit is said to be conditionally dispatchable if for each arrival sequence, there exists a stable as well as causal input-output map defined over the prefixes of the arrival sequence.

We show that an input-output map is causal if and only if it is representable as a *Mealy* automaton [10]. We also present a test for the stability of a causal input-output map using its Mealy automaton representation. It is clear that the stability of a dispatching unit implies its conditional stability. We show that the converse is also true, i.e., the two notions are equivalent. The notion of *semi-linear* sets [7, 24] is used to obtain a necessary and sufficient condition for the stability of a dispatching unit. It is also clear that the dispatchability of a dispatching unit implies its conditional dispatchability. We show the converse is not true. We obtain a necessary and sufficient condition for determining the conditional dispatchability of a given dispatching unit using concurrent composition [9] of systems. Finally, we obtain a necessary and sufficient condition for the dispatchability of a given dispatching unit.

The rest of the paper is organized as follows: Section 2 develops the notations used for the analysis. Section 3 defines the notions of dispatching units, dispatching policies, stability and dispatchability of dispatching units, and gives illustrative examples. Section 4-6 presents algorithmic tests for existence of respectively, stable, conditionally dispatchable, and dispatchable units. Techniques to synthesize dispatching policies whenever they exist are also presented in those sections. Section 7 discusses the conclusion and future research.

2 Notation and Preliminaries

As discussed in the introduction, an input language over a finite number of part types is used for describing the set of all possible sequences in which part types may arrive, and an output language is used for describing the set of all possible sequences in which they may depart. Let $n \in \mathcal{N}$ denote the total number of part types. The set Σ_I (respectively, Σ_O) is used to denote the set of events, each element of which corresponds to arrival (respectively, departure) of a certain part type. Thus $|\Sigma_I| = |\Sigma_O| = n$.

Given a finite set $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ of size $n \in \mathcal{N}$, Σ^* is used to denote the set of all possible finite sequences of elements belonging to Σ, including the zero length sequence ϵ; $\Sigma^{\leq N} \subseteq \Sigma^*$ is used to denote the set of sequences of length no greater than $N \in \mathcal{N}$. Given $s, t \in \Sigma^*$, we use $s \leq t$ to denote that s is a *prefix* of t. For a string $s \in \Sigma^*$ the notation $|s| \in \mathcal{N}$ is used to denote the length of s; and $\overline{s} \subseteq \Sigma^*$ denotes the set of all prefixes of s. For each $k \leq |s|$, the notation $s(k) \in \Sigma$ is used to denote the kth element of s, $s^{(k)} \leq s$ is used to denote the prefix of length k of s, and $s_{(k)} \in \Sigma^*$ denotes the suffix obtained by deleting the first k elements of s. For $s \in \Sigma^*$, and $k \leq n$, the notation $\#(\sigma_k, s)$ is used to denote the number of occurrences of the symbol σ_k in the string s; and the notation $\vec{s} \in \mathcal{N}^n$ is used to denote the *count vector* of s, also called the *score vector* or the *Parikh's map* [24] defined as:

$$\vec{s} := [\#(\sigma_1, s), \ldots, \#(\sigma_k, s), \ldots, \#(\sigma_n, s)].$$

For example if $s = \sigma_1\sigma_2\sigma_n\sigma_3\sigma_2$, then $\vec{s} = [1, 2, 1, 0, 0, \ldots, 0, 1]$. The count vector is used below for calculating the difference between the numbers of each part type in an arrival sequence and in the corresponding departure sequence under a given dispatching policy. A subset $L \subseteq \Sigma^*$ is called a *language* over Σ. Given $L \subseteq \Sigma^*$, $\overline{L}$ is used to denote (prefix) closure of L, i.e., $\overline{L} := \{t \in \Sigma^* \mid \exists s \in L \text{ s.t. } t \in \overline{s}\}$. The notation $s \equiv_L s'$ is used to denote that s, s' are Nerode equivalent with respect to L. The set of count vectors of $L \subseteq \Sigma^*$, denoted $\vec{L}$, is defined to be $\vec{L} := \{\vec{s} \in \mathcal{N}^n \mid s \in L\}$.

Letting I, O denote the input, output language constraints respectively, we have $I \subseteq \Sigma_I^*$, and $O \subseteq \Sigma_O^*$. It is clear that if a sequence $s \in I$ is an arrival sequence, and if $t \leq s$, then t should also be an arrival sequence. Hence we require that I be a closed language, i.e., $I = \overline{I}$. Similarly, we require that O be closed, i.e., $O = \overline{O}$. Both I and O can in general contain infinitely many sequences. However, if they are *regular* languages, then they can be concisely described using *deterministic finite automata* (DFA) (refer to [10] for the definition of regular languages).

A deterministic automaton (DA) G is defined to be the quadruple:

$$G := (Q, \Sigma, \delta, q^0),$$

where Q denotes the state set of G, Σ is the event set of G, $\delta : Q \times \Sigma \to Q$ denotes the (partial) deterministic transition function of G, and $q^0 \in Q$ denotes the initial state of G. G is a DFA if Q is finite. The transition function $\delta(\cdot, \cdot)$ is recursively extended to $\delta^* : Q \times \Sigma^* \to Q$ as follows:

$$\forall q \in Q, s \in \Sigma^* : \delta^*(q, s) := \delta^*(\delta(q, s(1)), s_{(1)}); \quad \delta^*(q, \epsilon) := q.$$

The language *generated* by G, denoted $L(G)$, is defined to be:

$$L(G) := \{s \in \Sigma^* \mid \delta^*(q^0, s) \text{ is defined}\}.$$

It follows from this definition that $L(G)$ is closed. A DA $G' := (Q, \Sigma, \delta', q^0)$ is said to be a *subautomaton* [10] of the DA G, written $G' \leq G$, if for each $q \in Q$ and $\sigma \in \Sigma$, $\delta'(q, \sigma) = \delta(q, \sigma)$ whenever it is defined.

Given $I \subseteq \Sigma_I^*, O \subseteq \Sigma_O^*$ as specifications for the input, output languages respectively, the DAs $G_I := (Q_I, \Sigma_I, \delta_I, q_I^0), G_O := (Q_O, \Sigma_O, \delta_O, q_O^0)$ are used as their respective generators; i.e., $L(G_I) = I, L(G_O) = O$.

n different buffers, one for each part type, are used for temporary storage. Let B_k denote the buffer for the kth part type, and $|B_k| \in \mathcal{N}$ denote its capacity. We use the notation

$\overrightarrow{|B|} \in \mathcal{N}^n$, called the *buffer capacity vector*, to denote the vector formed by considering the capacities of the individual buffers, i.e.,

$$\overrightarrow{|B|} := [|B_1|, |B_2|, \ldots, |B_k|, \ldots, |B_n|].$$

This system of n buffers can be described as a "buffer DFA", $G_B = (Q_B, \Sigma_B, \delta_B, q_B^0)$, where (i) $Q_B := \{\vec{b} \in \mathcal{N}^n \mid \vec{0} \leq \vec{b} \leq \overrightarrow{|B|}\}$[1] is the finite set of states—each state is a vector representing the number of buffered parts of each type, (ii) $\Sigma_B := \Sigma_I \times \Sigma_O^{\leq \|B\|}$, where $\|B\| := \sum_{k=1}^n |B_k|$ is the sum of all the buffer capacities—each transition represents a possible arrival event and a possible departure sequence at the indicated state, (iii) $q_B^0 := \vec{0}$—initially all the buffers are empty, and (iv) the transition function is defined as follows ($\vec{b} \in Q_B, \sigma \in \Sigma_I, s \in \Sigma_O^{\leq \|B\|}$):

$$\delta_B(\vec{b}, (\sigma, s)) := \begin{cases} [\vec{b} + \vec{\sigma} - \vec{s}] & \text{if } \vec{0} \leq [\vec{b} + \vec{\sigma} - \vec{s}] \leq \overrightarrow{|B|} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Thus when an arrival event $\sigma \in \Sigma_I$ and a departure sequence $s \in \Sigma_O^{\leq \|B\|}$ occur in the buffer state $\vec{b}$, then the resulting buffer state is obtained by adding the count vector $[\vec{\sigma} - \vec{s}]$ to the buffer state $\vec{b}$, provided the resulting buffer state satisfies the buffer capacity constraints.

Example 1 Consider for example a dispatching unit for three part types $\{a, b, c\}$. Suppose

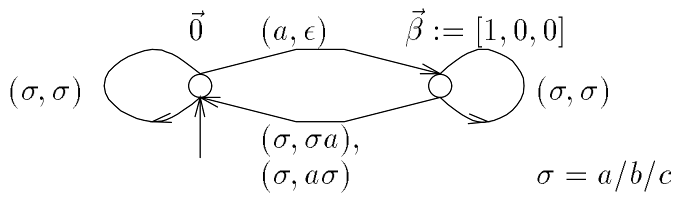


Figure 1: Buffer DFA with $\overrightarrow{|B|} = [1, 0, 0]$

there is a single buffer space for the part type a, and there is no buffer space for other part types, i.e., the buffer capacity vector $\overrightarrow{|B|} = [1, 0, 0]$. Then the corresponding buffer DFA G_B is shown in Figure 1. For future convenience, the notation $\vec{\beta} := [1, 0, 0]$ has been introduced to denote the "full" buffer state. ■

Let the map $\pi_I : \Sigma_I \times \Sigma_O^* \to \Sigma_I$ denote the projection operation onto the output events, i.e., $\pi_I(\sigma, s) := \sigma$, where $\sigma \in \Sigma_I$ and $s \in \Sigma_O^*$. This map can be extended to $\pi_I : 2^{(\Sigma_I \times \Sigma_O^*)^*} \to 2^{\Sigma_I^*}$ in a natural way. The map $\pi_O : 2^{(\Sigma_I \times \Sigma_O^*)^*} \to 2^{\Sigma_O^*}$ can be defined analogously. The following proposition can be obtained in a straightforward manner using induction on the length of strings.

[1] $\vec{0}$ denotes the zero vector of size n, and the "$\leq$" relation holds for vectors if it holds for their each respective components.

Proposition 1 Let $G_B = (Q_B, \Sigma_B, \delta_B, q_B^0)$ be the buffer DFA as defined above. Then

$$L(G_B) = \{s \in (\Sigma_I \times \Sigma_O^{\leq \|B\|})^* \mid \forall k \leq |s| : \vec{0} \leq [\overrightarrow{\pi_I(s^{(k)})} - \overrightarrow{\pi_O(s^{(k)})}] \leq \overrightarrow{|B|}\}.$$

Given a DA $G := (Q, \Sigma_I \times \Sigma_O^*, \delta, q^0)$, we use $\pi_I(G)$ to denote its "projection" onto the input events, which is the automaton obtained by replacing each transition label $(\sigma, s) \in \Sigma_I \times \Sigma_O^*$ of G by the label σ. The projection $\pi_O(G)$ is similarly defined. Note that the projection automaton $\pi_I(G)$ or $\pi_O(G)$ may be nondeterministic even when G is deterministic. Also, $L(\pi_I(G)) = \pi_I(L(G))$ and $L(\pi_O(G)) = \pi_O(L(G))$.

3 Dispatching Unit and Dispatching Policies

Having introduced the notation, we now formally define a dispatching unit, the notions of conditional stability, stability, conditional dispatchability, and dispatchability of a dispatching unit. Given an input language $I \subseteq \Sigma_I^*$, an output language $O \subseteq \Sigma_O^*$, and a buffer capacity vector $\overrightarrow{|B|} \in \mathcal{N}^n$, the corresponding *dispatching unit* is defined to be the triple $(I, O, \overrightarrow{|B|})$. We are mainly interested in stability and causality of input-output maps defined either over the input language, or over the prefixes of a given arrival sequence. However, in general, these can be defined over any closed sublanguage of the input language.

Definition 1 Given a dispatching unit $(I, O, \overrightarrow{|B|})$, and a closed language $I' \subseteq I$, an input-output map $\mathcal{D}_{I'} : I' \to O$ is said to be *stable* (or *input-output bounded*) *over* I' if

Stability: $\forall s \in I' : \vec{0} \leq [\vec{s} - \overrightarrow{\mathcal{D}_{I'}(s)}] \leq \overrightarrow{|B|}$.

It is said to be *causal over* I' if it is *prefix preserving*, i.e.,

Causality: $\forall s, t \in I' : s \leq t \Rightarrow \mathcal{D}_{I'}(s) \leq \mathcal{D}_{I'}(t)$.

In other words, the stability of an input-output map $\mathcal{D}_{I'} : I' \to O$ implies that for each arrival sequence $s \in I'$ the difference between the count vectors of s and $\mathcal{D}_{I'}(s)$ is (i) bounded below by the zero vector, implying that the number of dispatched parts does not exceed the number of arrived parts for each part type, and (ii) bounded above by the buffer capacity vector, implying that the buffers of each part type never overflow. The causality of an input-output map $\mathcal{D}_{I'} : I' \to O$ implies that the departure sequence corresponding to the arrival sequence $s \in I'$ is a prefix of the departure sequence corresponding to the arrival sequence $t \in I'$ whenever s is a prefix of t. Consequently, upon occurrence of an arrival event $\sigma \in \Sigma_I$ following an arrival sequence $s \in I'$ (so that $s\sigma \in I'$), the sequence of parts to be dispatched is obtained as $\mathcal{D}_{I'}(s\sigma)_{(|\mathcal{D}_{I'}(s)|)}$, i.e., it is the suffix obtained by deleting the initial prefix $\mathcal{D}_{I'}(s)$ from the sequence $\mathcal{D}_{I'}(s\sigma)$. This is well defined since $\mathcal{D}_{I'}(\cdot)$ is causal.

Remark 1 A stable and causal input-output map can be used to obtain a dispatching policy which determines the sequence of parts to be dispatched following each arrival sequence in such a manner that the output language constraint as well as the buffer capacity constraint is satisfied. More formally, given a dispatching unit $(I, O, \overrightarrow{|B|})$ and a closed language $I' \subseteq I$, a *dispatching policy over* I' is a map $\mathcal{P}_{I'} : I' \to \Sigma_O^*$ such that

P1: $\forall s \in I' : t_s := \mathcal{P}_{I'}(s^{(1)})\mathcal{P}_{I'}(s^{(2)}) \ldots \mathcal{P}_{I'}(s) \in O$, and

P2: $\forall s \in I' : \vec{0} \leq [\vec{s} - \vec{t_s}] \leq \overrightarrow{|B|}$.

Note that given a dispatching policy $\mathcal{P}_{I'} : I' \to \Sigma_O^*$ over I', it induces a stable and causal input-output map $\mathcal{D}_{I'} : I' \to O$ over I' defined as $\mathcal{D}_{I'}(s) := \mathcal{P}_{I'}(s^{(1)})\mathcal{P}_{I'}(s^{(2)}) \ldots \mathcal{P}_{I'}(s)$ for each $s \in I'$. Conversely, given a stable and causal input-output map $\mathcal{D}_{I'} : I' \to O$ over I', it induces a dispatching policy $\mathcal{P}_{I'} : I' \to \Sigma_O^*$ over I' defined as $\mathcal{P}_{I'}(s) := \mathcal{D}_{I'}(s)_{(|\mathcal{D}_{I'}(s^{(|s|-1)})|)}$ if $s \neq \epsilon$, and $\mathcal{P}_{I'}(\epsilon) := \epsilon$. ■

Definition 2 Given a dispatching unit $(I, O, \overrightarrow{|B|})$, it is said to be *stable* if there exists a stable input-output map over I; it is said to be *dispatchable* if there exists a stable and causal input-output map over I (equivalently, a dispatching policy over I).

Remark 2 Note that the notion of a "causal" dispatching unit defined as the existence of a causal input-output map over I is not of any interest, as given a dispatching unit $(I, O, \overrightarrow{|B|})$, there exists a trivial causal input-output map $\mathcal{D}_I : I \to O$ defined as $\mathcal{D}_I(s) := \epsilon$, for each $s \in I$. Hence, we have only defined the notions of stability and dispatchability (and not "causality") of a dispatching unit. ■

The existence of a dispatching policy over I requires that the sequence of parts to be dispatched after an arrival sequence $s \in I$ should depend only on s—the past sequence of arrivals, and no knowledge about the future arrival sequence is assumed. However, as discussed in the introduction, in some applications the entire arrival sequence may be known in advance, and a dispatching policy should be determined by taking this fact into account. Hence, we define the weaker notions of conditional stability and conditional dispatchability.

Definition 3 Given a dispatching unit $(I, O, \overrightarrow{|B|})$, it is said to be *conditionally stable* if for each $s \in I$, there exists a stable input-output map over $\overline{s}$; it is said to be *conditionally dispatchable* if for each $s \in I$, there exists a stable and causal input-output map over $\overline{s}$ (equivalently, a dispatching policy over $\overline{s}$).

Thus conditional stability requires existence of a set of input-output maps $\{\mathcal{D}_{\overline{s}} : \overline{s} \to O \mid s \in I\}$ such that each map $\mathcal{D}_{\overline{s}}(\cdot)$ is stable, and conditional dispatchability requires that each such map be also causal. It follows from Definition 2 that dispatchability of a dispatching unit implies its stability; similarly it follows from Definition 3 that conditional dispatchability of a dispatching unit implies its conditional stability. It also follows from Definitions 2 and

3 that stability of a dispatching unit implies its conditional stability, since given a stable input-output map $\mathcal{D}_I : I \to O$ and any sequence $s \in I$, restriction of $\mathcal{D}_I(\cdot)$ to $\overline{s}$ yields a stable input-output map over $\overline{s}$; similarly dispatchability of a dispatching unit implies its conditional dispatchability. Note that given a set of stable input-output maps $\{\mathcal{D}_{\overline{s}} : \overline{s} \to O \mid s \in I\}$, one can obtain a stable input-output map $\mathcal{D}_I : I \to O$ by defining $\mathcal{D}_I(s) := \mathcal{D}_{\overline{s}}(s)$ for each $s \in I$. Hence conditional stability of a dispatching unit also implies its stability. Thus the two notions are equivalent. However, as illustrated by an example below (refer to Example 4), conditional dispatchability of a dispatching unit does not imply its dispatchability. Thus, there are essentially three classes of dispatching units—stable, conditionally dispatchable, and dispatchable. Next we illustrate this using a series of examples. The first example illustrates that not all dispatching units are stable, and the class of stable dispatching units is nonempty.

Example 2 Let the input, output language constraint for two part types a and b be given as: $I = \{\overline{a^k b^k} \mid k \geq 0\}$, and $O = \{\overline{b^k a^k} \mid k \geq 0\}$. Then there exists no finite buffer capacity vector $\overrightarrow{|B|} < [\infty, \infty]$ such that $(I, O, \overrightarrow{|B|})$ is stable. In order to see this suppose $\overrightarrow{|B|} = [n_1, n_2] \in \mathcal{N}^2$, and consider the input string $a^{n_1+1} \in I$. Note that due to the output language constraint, any dispatcher before dispatching a certain number of parts of type a must dispatch an equal or greater number of parts of type b. Since no part of type b is present in a^{n_1+1}, a dispatcher is forced to buffer all the $n_1 + 1$ parts of type a. However, this violates the buffer capacity constraint. This illustrates that not every dispatching unit is stable.

Consider a different setting in which $I = \{\overline{a^k b^k} \mid k \geq 0\}$, $O = \{\overline{a^{2k} b^{2k}} \mid k \geq 0\}$, and let $\overrightarrow{|B|} = [1, 1]$. Then an input-output map $\mathcal{D}_I : I \to O$ defined as $\mathcal{D}_I(a^k b^l) := a^{2\lfloor \frac{k}{2} \rfloor} b^{2\lfloor \frac{l}{2} \rfloor}$ for each $k \geq l \geq 0$ is a stable input-output map, where $\lfloor \frac{k}{2} \rfloor$ denotes the largest integer smaller than $\frac{k}{2}$. Hence, $(I, O, \overrightarrow{|B|})$ is stable. This illustrates that the class of stable dispatching units is nonempty. ■

The next example illustrates that the class of conditionally dispatchable dispatching units is a proper subclass of the class of stable dispatching units.

Example 3 Let the input and output language constraints for the two part types a and b be given as $I = \overline{(ab)^*}$ and $O = a^* b^*$. Then an input-output map $\mathcal{D}_I : I \to O$ defined as $\mathcal{D}_I((ab)^k) := a^k b^k$ and $\mathcal{D}_I((ab)^k a) := a^{k+1} b^k$ for each $k \geq 0$ is a stable input-output map. Thus $(I, O, \overrightarrow{|B|})$ is stable with $\overrightarrow{|B|} = [0, 0]$. However, $(I, O, \overrightarrow{|B|})$ is not conditionally dispatchable for any $\overrightarrow{|B|} < [\infty, \infty]$. In order to see this, suppose the buffer capacity vector is $\overrightarrow{|B|} = [n_1, n_2] \in \mathcal{N}^2$ and let $n := \max\{n_1, n_2\}$. Consider the input string $s := (ab)^{2n+2} \in I$ and its prefix $t := (ab)^{n+1}$. Then for the buffer capacity constraint to hold, the output string for s must of the form $a^{2n+2-n_1} a^k b^{2n+2-n_2} b^l$, where $k \leq n_1$ and $l \leq n_2$. On the other hand, the output string for t must be of the form $a^{n+1-n_1} a^{k'} b^{n+1-n_2} b^{j'}$, where $k' \leq n_1$ and $j' \leq n_2$.

Note that $2n + 2 - n_1 + k \geq n_1 + 2 > n_1 + 1 \geq n + 1 - n_1 + k'$, and $n + 1 - n_2 + j' \geq 0$. Thus although t is a prefix of s, the output string for t cannot be a prefix of the output string for s. This shows that $(I, O, \overrightarrow{|B|})$ is not conditionally dispatchable. Thus the class of conditionally dispatchable dispatching units is a proper subclass of the class of stable ones.■

Finally, the following example illustrates that the class of dispatchable dispatching units is nonempty, and it is a proper subclass of the class of conditionally dispatchable ones.

Example 4 Let a, b, c, d represent the part types, and let the input language constraint be $I = \overline{(ab)^*c^*}$, and the output constraint be $O = \overline{(ba)^*c^*}$. Then a dispatching policy (i.e., a stable and causal input-output map) can be obtained as follows: buffer the initially arrived part a, and then repeatedly dispatch ba for each arrival of ba until the first arrival of part type c occurs, upon when dispatch bca (using the last arrived part b, buffered part a, and the currently arrived part c), and then continue dispatching part c whenever it arrives. This illustrates that the class of dispatchable dispatching units is nonempty.

Now, if the input and output language constraints are changed to $I = \overline{(ab)^*(c^* + d^*)}$, and $O = \overline{(ba)^*c^* + (ab)^*d^*}$, respectively, then a dispatching policy cannot be constructed. This is because the departure sequence must contain a prefix $ab \ldots ab$ if the arrival sequence contains the event d, whereas, it must contain a prefix $ba \ldots ba$ if the arrival sequence contains the event c. However, since the knowledge of the entire arrival sequence is not available in advance, any dispatcher is forced to buffer an unbounded number of parts. On the other hand, if the arrival sequence is known in advance, then it is easy to construct a dispatching policy. This illustrates that the class of dispatchable dispatching units is a proper subclass of the class of conditionally dispatchable dispatching units. ■

Figure 2 summarizes the relationship among various types of dispatching units defined above.

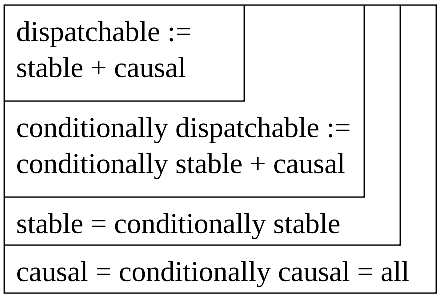


Figure 2: Relationship among various types of dispatching units

4 Stable Units

In this section we provide an existence condition for stable dispatching units and discuss a technique for verifying it. This requires the concept of semi-linear sets.

The motivation for studying semi-linear sets comes from the notion of count vector introduced in section 2. Let $\vec{x} = [x_1, x_2, \ldots, x_n]$ and $\vec{y} = [y_1, y_2, \ldots, y_n]$ be elements of $\mathcal{N}^n$, where for each $1 \leq k \leq n$, $x_k, y_k \in \mathcal{N}$. We define addition and scalar multiplication in $\mathcal{N}^n$ in the usual manner:

$$\vec{x} + \vec{y} := [x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n]; \quad \alpha\vec{x} := [\alpha x_1, \alpha x_2, \ldots, \alpha x_n],$$

where $\alpha \in \mathcal{N}$.

Definition 4 Given $\vec{c}, \vec{p_1}, \ldots, \vec{p_k} \in \mathcal{N}^n$, the set

$$X[\vec{c}; \vec{p_1}, \ldots, \vec{p_k}] := \{\vec{c} + n_1\vec{p_1} + \ldots + n_k\vec{p_k} \mid n_j \in \mathcal{N}, \forall j \leq k\} \subseteq \mathcal{N}^n$$

is called a *linear* set. $\vec{c}$ is the *constant*, and for each $j \leq k$, $\vec{p_j}$ is a *period* of the linear set. A subset of $\mathcal{N}^n$ is called a *semi-linear* set if it is a finite union of linear sets.

For example, the linear set $X[[10,10]; [1,0], [0,1]]$ represents the set of all points in a plane having integer coordinates greater than or equal to 10.

Semi-linear sets form an important class of subsets of $\mathcal{N}^n$ from the viewpoint of formal language theory because the count vector of any context-free language is a semi-linear set [24]. This fact was first shown by Parikh [24]. We prove a special case of this result for regular languages. Our proof has the advantage that it provides a decomposition of a regular language into a finite number of regular sublanguages whose count vectors are linear sets. Our method uses the regular expression representation of regular languages [10]. A *regular expression* defined on an alphabet set Σ is any expression containing the symbols: choice operation (represented by $+$), concatenation operation (represented by .), Kleene closure operation (represented by *), ϵ, and symbols from Σ.

Definition 5 A regular expression is called a *linear regular expression* if its count vector is a linear subset of $\mathcal{N}^n$.

An algebraic characterization of linear regular expressions is given next. Let $r^\sharp := r.r^*$, in which r is repeated any finite number of times but at least once. A regular expression can be written using the operations $\{+, ., ^\sharp\}$ by eliminating the operation * by applying the identity $r^* = r^\sharp + \epsilon$.

Lemma 1 Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a finite alphabet set of size $n \in \mathcal{N}$. Any expression composed of "$\sharp$", ".", ϵ, and any symbol from Σ is a linear regular expression.

Proof: Note that a regular expression composed of either ϵ or a single symbol from Σ is a linear regular expression. The linear set corresponding to ϵ equals $X[\vec{0};]$; and the linear set corresponding to a symbol $\sigma_k \in \Sigma$, where $k \leq n$, equals $X[[0, \ldots, 1, \ldots, 0];]$, in which the kth entry of the constant is 1. We need to show that the operations "." and "$\sharp$" preserve linearity.

We claim that if r_1, r_2 are linear regular expressions, then so are $r_1.r_2$ and $r_1^\sharp$. To see this let $\vec{r_1} = X[\vec{c}; \vec{p_1}, \ldots, \vec{p_k}]$ and $\vec{r_2} = X[\vec{d}; \vec{q_1}, \ldots, \vec{q_l}]$, where $\vec{c}, \vec{d} \in \mathcal{N}^n$ are constants, and

$\vec{p_i}, \vec{q_j} \in \mathcal{N}^n$ for each $i \leq k, j \leq l$ are periods. Then $\overrightarrow{r_1.r_2} = X[\vec{c}+\vec{d}; \vec{p_1}, \ldots, \vec{p_k}, \vec{q_1}, \ldots, \vec{q_l}]$, and $\overrightarrow{r_1^\sharp} = X[\vec{c}; \vec{c}, \vec{p_1}, \ldots, \vec{p_k}]$. ∎

We now present a method for decomposing a regular expression into a finite number of linear regular expressions.

Lemma 2 Let r be a regular expression. Then it can be written as $r_1 + r_2 + \ldots + r_n$, where r_k for each $k \leq n$ is a linear regular expression.

Proof: We first replace each occurrence of "$*$" by "$\sharp$" in r using the identity described above so that r contains only the operation symbols ".", "$\sharp$", and "$+$". Then the result follows from Lemma 1 and by noting these two facts: (i) Given any three regular expressions r_1, r_2, r_3 the following holds:

$$(r_1 + r_2).r_3 = r_1.r_3 + r_2.r_3 \text{ and } r_3.(r_1 + r_2) = r_3.r_1 + r_3.r_2,$$

and (ii) Given any two regular expressions r_1, r_2 the following holds:

$$(r_1 + r_2)^\sharp = r_1^\sharp + r_2^\sharp + (r_1^\sharp.r_2^\sharp)^\sharp + (r_2^\sharp.r_1^\sharp)^\sharp + (r_1^\sharp.r_2^\sharp)^\sharp.r_1^\sharp + (r_2^\sharp.r_1^\sharp)^\sharp.r_2^\sharp$$

The right hand side of the identity corresponds to the following choices between r_1 and r_2:

1. one or more of strings from r_1
2. one or more of strings from r_2
3. starts with a string in r_1, ends in a string in r_2
4. starts with a string in r_2, ends in a string in r_1
5. starts with a string in r_1, ends in a string in r_1, at least one string from r_2
6. starts with a string in r_2, ends in a string in r_2, at least one string from r_1

It is easily verified that these are the only possible choices. ∎

Since $\overrightarrow{r_1 + r_2} = \vec{r_1} \cup \vec{r_2}$ for any two regular expressions r_1 and r_2, the following Proposition is immediate from Lemmas 2 and 1.

Proposition 2 If L is a regular language over a finite set Σ, then its count vector $\vec{L} \subseteq \mathcal{N}^{|\Sigma|}$ is a semi-linear set.

Example 5 Consider the language a^*b^* defined over the alphabet set $\{a, b\}$. Then

$$a^*b^* = (a^\sharp + \epsilon).(b^\sharp + \epsilon) = (a^\sharp.b^\sharp + a^\sharp + b^\sharp + \epsilon).$$

The corresponding linear sets are:

$$\vec{\epsilon} = X[[0,0];]; \ \overrightarrow{a^\sharp} = X[[1,0];[1,0]]; \ \overrightarrow{b^\sharp} = X[[0,1];[0,1]]; \overrightarrow{a^\sharp.b^\sharp} = X[[1,1];[0,1],[1,0]].$$

∎

Using the tools of semi-linear sets, we next obtain conditions under which a given dispatching unit $(I, O, \overrightarrow{|B|})$ is stable, so that there exists a stable input-output map for it.

Consider two sets $P, Q \subseteq \mathcal{N}^n$ and define

$$P \oplus Q := \{\vec{w} \in \mathcal{N}^n \mid \exists \vec{u} \in P, \vec{v} \in Q \text{ s.t. } \vec{w} = \vec{u} + \vec{v}\}.$$

Note that if P, Q are semi-linear sets, then so is $P \oplus Q$. Define $\vec{B} := \{\vec{b} \in \mathcal{N}^n \mid \vec{0} \leq \vec{b} \leq \overrightarrow{|B|}\}$, i.e., $\vec{B}$ is the collection of all possible buffer state vectors. Since $\vec{B}$ is a finite set, it is clear that it is a semi-linear set.

Theorem 1 A dispatching unit $(I, O, \overrightarrow{|B|})$ is stable if and only if $\vec{I} \subseteq \vec{O} \oplus \vec{B}$.

Proof: $(\Rightarrow)$ Assume first that $(I, O, \overrightarrow{|B|})$ is stable. Consider any string $s \in I$ so that $\vec{s} \in \vec{I}$. Since $(I, O, \overrightarrow{|B|})$ is stable, there exists a map $\mathcal{D}_I : I \to O$ such that $\vec{0} \leq [\vec{s} - \overrightarrow{\mathcal{D}_I(s)}] \leq \overrightarrow{|B|}$. This implies that $[\vec{s} - \overrightarrow{\mathcal{D}_I(s)}] \in \vec{B}$, i.e., there exists $\vec{b} \in \vec{B}$ such that $[\vec{s} - \overrightarrow{\mathcal{D}_I(s)}] = \vec{b}$. This can be rewritten to obtain $\vec{s} = \overrightarrow{\mathcal{D}_I(s)} + \vec{b}$. Now since $\mathcal{D}_I(s) \in O$ and $\vec{b} \in \vec{B}$ we obtain that $\vec{s} \in \vec{O} \oplus \vec{B}$, as desired.

$(\Leftarrow)$ Assume next that $\vec{I} \subseteq \vec{O} \oplus \vec{B}$. Consider $s \in I$, then $\vec{s} \in \vec{I}$. Hence there exists $t \in O$ and $\vec{b} \in \vec{B}$ such that $\vec{s} = \vec{t} + \vec{b}$, i.e., $\vec{s} - \vec{t} = \vec{b}$. Since $\vec{b} \in \vec{B}$, we have that $\vec{0} \leq \vec{b} \leq \overrightarrow{|B|}$. Hence $\vec{0} \leq [\vec{s} - \vec{t}] \leq \overrightarrow{|B|}$. Thus by setting $\mathcal{D}_I(s) := t$, we can conclude that $(I, O, \overrightarrow{|B|})$ is stable. ∎

Remark 3 It follows from Theorem 1 and Proposition 2 that when I and O are regular languages, then the task of determining whether a given dispatching unit $(I, O, \overrightarrow{|B|})$ is stable is reduced to checking whether a semi-linear set is contained in another semi-linear set. It is shown in [7] that the containment problem for semi-linear sets in a general setting is *decidable*. ∎

Example 6 Consider the setting of Example 2 with $I = \{\overline{a^k b^k} \mid k \geq 0\}; O = \{\overline{a^{2k} b^{2k}} \mid k \geq 0\}; \overrightarrow{|B|} = [1, 1]$. Then

$$\begin{aligned}
\vec{I} &= \{[n_1, n_2] \mid 0 \leq n_1 - n_2 \leq 1\}, \\
\vec{O} &= \{[n_1, n_2] \mid 0 \leq n_1 - n_2 \leq 2, n_1 \text{ or } n_2 \text{ even}\}, \\
\vec{B} &= \{[n_1, n_2] \mid 0 \leq n_1, n_2 \leq 1\}.
\end{aligned}$$

Hence it follows that

$$\vec{O} \oplus \vec{B} = \{[n_1, n_2] \mid -1 \leq n_1 - n_2 \leq 3\}.$$

Thus $\vec{I} \subseteq \vec{O} \oplus \vec{B}$, so Theorem 1 implies that the triple $(I, O, \overrightarrow{|B|})$ is a stable dispatching unit. ∎

Theorem 1 demonstrates that the property of stability only depends on the count vectors of the languages involved. Given a finite set Σ, define an equivalence relation $\cong\subseteq 2^{\Sigma^*}\times 2^{\Sigma^*}$ such that for $K, L\subseteq\Sigma^*$, $K\cong L$ if $\vec{K}=\vec{L}$. In this case K and L are said to be *count equivalent* or *letter equivalent*. A consequence of Theorem 1 is the following corollary:

Corollary 1 Let $I_1, I_2\subseteq\Sigma_I^*$, $O_1, O_2\subseteq\Sigma_O^*$ be such that $I_1\cong I_2$ and $O_1\cong O_2$; then $(I_1, O_1, \overrightarrow{|B|})$ is stable if only if $(I_2, O_2, \overrightarrow{|B|})$ is stable.

5 Conditionally Dispatchable Units

In this section we obtain a necessary and sufficient condition under which a given dispatching unit $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable, so that for each $s\in I$, there exists a stable and causal input-output map defined over prefixes of s. The condition is obtained in terms of the *output-composition* of the buffer DFA G_B and the DA G_O which generates the output language O.

Definition 6 The *output-composition* of G_B and G_O, denoted G_{BO}, is another DA $G_{BO}:=(Q_B\times Q_O, \Sigma_B, \delta_{BO}, (\vec{0}, q_O^0))$, where the transition function is defined as ($\vec{b}\in Q_B, q_O\in Q_O, \sigma\in\Sigma_I, s\in\Sigma_O^{\leq\|B\|}$):

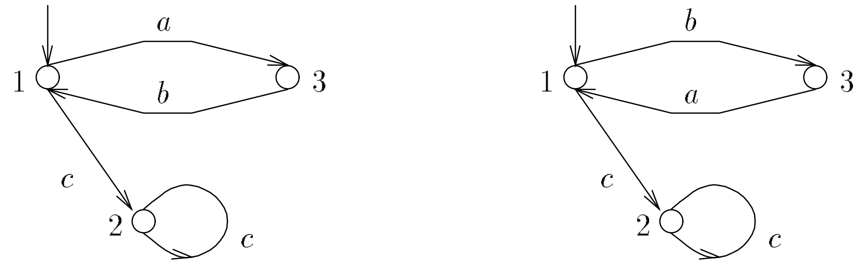
$$\delta_{BO}((\vec{b}, q_O), (\sigma, s)) := \begin{cases} (\delta_B(\vec{b}, (\sigma, s)), \delta_O^*(q_O, s)) & \text{if } \delta_B(\vec{b}, (\sigma, s)), \delta_O^*(q_O, s) \text{ defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Thus a transition corresponding to an arrival event $\sigma\in\Sigma_I$ and a departure sequence $s\in\Sigma_O^{\leq\|B\|}$ is defined in a buffer state $\vec{b}$ and a state q_O of the generator for O if and only if (i) the resulting buffer state $[\vec{b}+\vec{\sigma}-\vec{s}]$ is an admissible buffer state (i.e., buffer capacity constraints are satisfied), and (ii) the departure sequence s is executable in the state q_O of G_O (i.e., s is a valid extension of the departure sequence, execution of which from the initial state q_O^0 leads to the state q_O in G_O). Since the number of states of the buffer DFA equals $\prod_{k=1}^n(|B_k|+1)$, the number of states in G_{BO} equals $|Q_O|\cdot[\prod_{k=1}^n(|B_k|+1)]$.

Example 7 Consider the setting of Example 4 with $I=\overline{(ab)^*c^*}$; $O=\overline{(ba)^*c^*}$; $\overrightarrow{|B|}=[1,0,0]$. The DFAs G_I and G_O generating the input and output languages, respectively, are shown in Figure 3(a) and 3(b), respectively; the buffer DFA G_B with capacity vector $\overrightarrow{|B|}=[1,0,0]$ is shown in Figure 1.

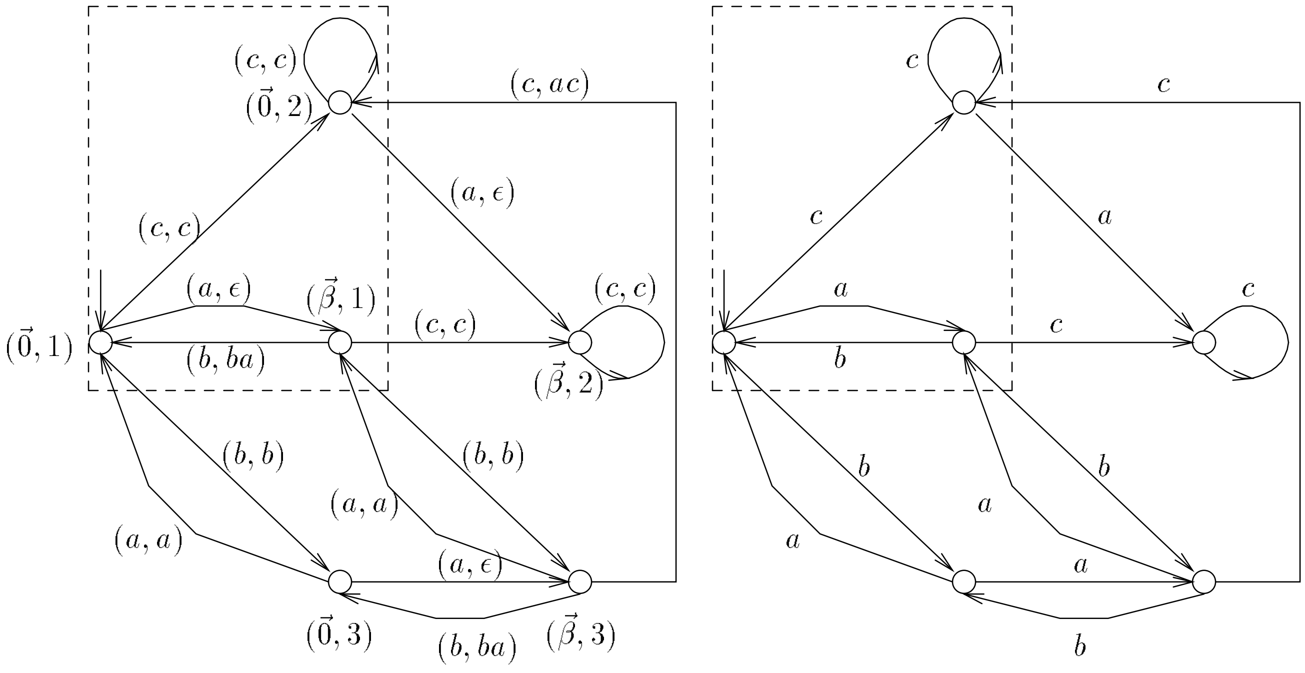
Figure 4(a) depicts the DFA G_{BO} obtained by output-composition of G_B and G_O as stated in Definition 6. ■

The result of the following proposition can be obtained in a straightforward manner by using induction on the length of strings.



(a) (b)

Figure 3: Diagram illustrating DFAs G_I and G_O



(a) (b)

Figure 4: Diagram illustrating G_{BO} and $\pi_I(G_{BO})$

Proposition 3 Consider the DA $G_{BO} = (Q_B \times Q_O, \Sigma_B, \delta_{BO}, (\vec{0}, q_O^0))$ as in Definition 6. Then

$$L(G_{BO}) = \{s \in L(G_B) \mid \pi_O(s) \in O\}.$$

It follows from Proposition 3 that $L(G_{BO}) \subseteq L(G_B)$ and $\pi_O(L(G_{BO})) \subseteq O$.

Theorem 2 A dispatching unit $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable if and only if $I \subseteq \pi_I(L(G_{BO}))$.

Proof: ($\Rightarrow$) Assume first that $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable. Pick any string $s \in I$; then we need to show that $s \in \pi_I(L(G_{BO}))$. Since the dispatching unit $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable, it follows from Definition 3 that there exists a map $\mathcal{D}_{\overline{s}} : \overline{s} \to O$ that is stable and causal over all prefixes of s. Recall that for each $k \leq |s|$, the notation $s(k) \in \Sigma_I$ is used to denote the kth element in s; the notation $s^{(k)} \leq s$ is used to denote the prefix of length k of s; and the notation $s_{(k)} \in \Sigma_I^*$ is used to denote the sequence obtained by deleting the initial k symbols from s. Consider the following sequence of pairs of arrival events and departure sequences:

$$s_{IO} := \{s(1), \mathcal{D}_{\overline{s}}(s^{(1)})\} \ldots \{s(k), \mathcal{D}_{\overline{s}}(s^{(k)})_{(|\mathcal{D}_{\overline{s}}(s^{(k-1)})|)}\} \ldots \{s(|s|), \mathcal{D}_{\overline{s}}(s)_{(|\mathcal{D}_{\overline{s}}(s^{(|s|-1)})|)}\} \in (\Sigma_I \times \Sigma_O^*)^*,$$

where the kth pair consists of the arrival event $s(k) \in \Sigma_I$ and the departure sequence obtained by deleting the initial prefix $\mathcal{D}_{\overline{s}}(s^{(k-1)})$ from the departure sequence $\mathcal{D}_{\overline{s}}(s^{(k)})$. Note that this is well defined since the map $\mathcal{D}_{\overline{s}}(\cdot)$ is causal over prefixes of s. Since the map $\mathcal{D}_{\overline{s}}(\cdot)$ is also stable over the prefixes of s, the following holds for each $k \leq |s|$: $\vec{0} \leq [\overrightarrow{s^{(k)}} - \overrightarrow{\mathcal{D}_{\overline{s}}(s^{(k)})}] \leq \overrightarrow{|B|}$. Since $s^{(k)} = \pi_I(s_{IO}^{(k)})$ and $\mathcal{D}_{\overline{s}}(s^{(k)}) = \pi_O(s_{IO}^{(k)})$, we obtain that for each $k \leq |s|, \vec{0} \leq [\overrightarrow{\pi_I(s_{IO}^{(k)})} - \overrightarrow{\pi_O(s_{IO}^{(k)})}] \leq \overrightarrow{|B|}$. Hence it follows from Proposition 1 that $s_{IO} \in L(G_B)$. It is clear that $\pi_O(s_{IO}) = \mathcal{D}_{\overline{s}}(s)$, hence $\pi_O(s_{IO}) \in O$. Consequently, it follows from Proposition 3 that $s_{IO} \in L(G_{BO})$. Since $\pi_I(s_{IO}) = s$, we obtain as desired that $s \in \pi_I(L(G_{BO}))$.

($\Leftarrow$) Next assume that $I \subseteq \pi_I(L(G_{BO}))$. Pick $s \in I$, then we need to show that there exists a map $\mathcal{D}_{\overline{s}} : \overline{s} \to O$ which is stable and causal over all prefixes of s. Since $I \subseteq \pi_I(L(G_{BO}))$, there exists a string $s_{IO} \in L(G_{BO})$ such that $\pi_I(s_{IO}) = s$. For a length k prefix of s define $\mathcal{D}_{\overline{s}}(s^{(k)}) := \pi_O(s_{IO}^{(k)})$. Since $s_{IO} \in L(G_{BO}) \subseteq L(G_B)$ and $s_{IO}^{(k)}$ is the length k prefix of s_{IO}, we have that $s_{IO}^{(k)} \in L(G_B)$. Hence it follows from Proposition 1 that $\vec{0} \leq [\overrightarrow{\pi_I(s_{IO}^{(k)})} - \overrightarrow{\pi_O(s_{IO}^{(k)})}] \leq \overrightarrow{|B|}$. Since $\pi_I(s_{IO}^{(k)}) = s^{(k)}$ and $\pi_O(s_{IO}^{(k)}) = \mathcal{D}_{\overline{s}}(s^{(k)})$, we obtain that $\vec{0} \leq [\overrightarrow{s^{(k)}} - \overrightarrow{\mathcal{D}_{\overline{s}}(s^{(k)})}] \leq \overrightarrow{|B|}$. Since k is arbitrary, this implies that the map $\mathcal{D}_{\overline{s}}(\cdot)$ is stable over all prefixes of s. It is clear that if $k \leq l$, then $\mathcal{D}_{\overline{s}}(s^{(k)}) = \pi_O(s_{IO}^{(k)}) \leq \pi_O(s_{IO}^{(l)}) = \mathcal{D}_{\overline{s}}(s^{(l)})$. This proves that the map $\mathcal{D}_{\overline{s}}(\cdot)$ is also causal over all prefixes of s. ∎

Remark 4 It follows from Theorem 2 that the conditional dispatchability, i.e., existence of a stable and causal input-output map over all prefixes of a known arrival sequence for the given dispatching unit $(I, O, \overrightarrow{|B|})$, can be determined by testing whether the input language I is contained in the language obtained by projecting the language of the output-composed DA G_{BO} onto the input event set. Also note that while the statement of Theorem 2 provides an *existence* condition for a stable and conditionally causal dispatching policy, its proof is *constructive*, i.e., it provides a technique to compute such a dispatching policy whenever it exists. (Refer to the example below for illustration of such a construction.) ■

Example 8 Consider the setting of Example 7. As mentioned above, the corresponding G_I and G_O are shown in Figures 3(a) and 3(b), respectively, G_B is shown in Figure 1, and G_{BO} is shown in Figure 4(a). Figure 4(b) depicts the automaton $\pi_I(G_{BO})$. Consider the sub-automaton of $\pi_I(G_{BO})$ lying within the dashed rectangular area as shown in Figure 4(b). Then the language generated by this sub-automaton equals $\overline{(ab)^*c^*} = I$. Hence we conclude that $I \subseteq \pi_I(L(G_{BO}))$. Thus it follows from Theorem 2 that the corresponding $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable.

The corresponding dispatching policy can be obtained as follows: Pick any arrival sequence $s \in I$. Since $I \subseteq \pi_I(L(G_{BO}))$, the arrival sequence s can be traced in $\pi_I(G_{BO})$ starting from the initial state. Consider the corresponding path in the DFA G_{BO}. Then the required dispatching policy is obtained simply by considering the departure sequence corresponding to each arrival event in the sequence s. Suppose for example the arrival sequence is abc. Then upon arrival of a, nothing is output (a is buffered, this is feasible as $\overrightarrow{|B|} = [1, 0, 0]$); upon arrival of next b, ba is output; and finally, upon arrival of c, c is output. ■

6 Dispatchable Units

In this section we obtain conditions under which a given dispatching unit $(I, O, \overrightarrow{|B|}\ )$ is dispatchable so that a stable and causal input-output map or a dispatching policy exists for it. We first show that a causal input-output map can be represented as a *Deterministic Mealy Automaton* (DMA) [10]. A condition for stability of a given causal input-output map is then obtained using its DMA representation.

Letting M denote a DMA, it is a six-tuple:

$$M := (Q, \Sigma, \Delta, \delta, \theta, q^0),$$

where Q, Σ, δ, q^0 are as in a DA, Δ is the *output event* set, and $\theta : Q \times \Sigma \to \Delta^*$ is the deterministic output function. M is said to be a deterministic finite Mealy Automata (DFMA) if Q is finite. The output function is extended to $\theta^* : Q \times \Sigma^* \to \Delta^*$ recursively as follows:

$$\forall q \in Q, s \in \Sigma^* : \theta^*(q, s) := \theta^*(\theta(q, s(1)), s_{(1)}); \quad \theta^*(q, \epsilon) := \epsilon.$$

The *input language*, denoted $L_I(M) \subseteq \Sigma^*$, and the *output language*, denoted $L_O(M) \subseteq \Delta^*$, of M are defined as:

$$L_I(M) := \{s \in \Sigma^* \mid \delta^*(q^0, s) \text{ defined}\}; \quad L_O(M) := \{t \in \Delta^* \mid \exists s \in L_I(M) \text{ s.t. } \theta^*(q^0, s) = t\}.$$

A DMA is a DA together with an output function which describes the outputs of the DA on its state transitions. Conversely, a DMA can be viewed as a DA—similar in structure to the buffer DFA G_B, which we describe next. Given a DMA $M := (Q, \Sigma, \Delta, \delta, \theta, q^0)$, it can equivalently be represented as a DA $G_M := (Q, \Sigma \times \Delta^*, \delta_M, q^0)$, where the transition function $\delta_M : Q \times (\Sigma \times \Delta^*) \to Q$ is defined as follows ($q \in Q, \sigma \in \Sigma, s \in \Delta^*$):

$$\delta_M(q, (\sigma, s)) := \begin{cases} \delta(q, \sigma) & \text{if } \delta(q, \sigma) \text{ defined, and } \theta(q, \sigma) = s \\ \text{undefined} & \text{otherwise} \end{cases}$$

Note that $L(G_M) \subseteq (\Sigma \times \Delta^*)^*$. It is easily seen that $\pi_I(L(G_M)) = L_I(M)$ and $\pi_O(L(G_M)) = L_O(M)$.

Definition 7 Given a DMA $M := (Q, \Sigma_I, \Sigma_O, \delta, \theta, q^0)$, an input-output map $\mathcal{D}_I : I \to O$ is said to be *realizable as* M, if $I \subseteq L_I(M), L_O(M) \subseteq O$, and $\mathcal{D}_I(s) = \theta^*(q^0, s)$. for each $s \in I$. In this case, the M is said to *realize* the input-output map $\mathcal{D}_I(\cdot)$.

Note that $\mathcal{D}_I(s) = \theta^*(q^0, s)$ is defined for each $s \in I$ since (i) $\theta^*(q^0, s)$ is defined for each $s \in L_I(M)$ and $I \subseteq L_I(M)$, and (ii) $\theta^*(q^0, s) \in L_O(M) \subseteq O$. The set of maps realizable as DFMAs is the set of regular causal input-output maps, which we define next.

Definition 8 Given a causal input-output map $\mathcal{D}_I : I \to O$, it is said to be *regular causal* if the equivalence relation $\simeq \subseteq \Sigma_I^* \times \Sigma_I^*$ defined as ($s, s' \in \Sigma_I^*$):

$$s \simeq s' \iff \forall t \in \Sigma_I^* : [st \in I \Leftrightarrow s't \in I] \wedge [st, s't \in I \Rightarrow \mathcal{D}_I(st)_{(|\mathcal{D}_I(s)|)} = \mathcal{D}_I(s't)_{(|\mathcal{D}_I(s')|)}]$$

is of finite index.

Note that in Definition 8 if $st, s't \in I$, then $s, s' \in I$ since I is closed. Hence $\mathcal{D}_I(s), \mathcal{D}_I(s')$, $\mathcal{D}_I(st), \mathcal{D}_I(s't)$ are all well defined. Also, note that if $s, s' \notin I$, then there exists no sequence $t \in \Sigma_I^*$ such that $st, s't \in I$. Hence if $s, s' \notin I$, then $s \simeq s'$. Note that the equivalence relation $\simeq$ given in Definition 8 is finer than the *Nerode equivalence* defined in theory of formal languages [10, 11]. A proof similar to the Nerode equivalence based proof, which establishes equivalence of regular languages and DFAs [10], can be used to obtain the following result:

Theorem 3 An input-output map $\mathcal{D}_I : I \to O$ is causal (respectively, regular causal) if and only if it is realizable as a DMA (respectively, DFMA) $M := (Q, \Sigma_I, \Sigma_O, \delta, \theta, q^0)$ with $I \subseteq L_I(M)$ and $L_O(M) \subseteq O$.

Proof: We only prove the second claim; the first claim can be proved analogously.

($\Rightarrow$) Assume that a regular causal input-output map $\mathcal{D}_I : I \to O$ is given. We need to show that there exists a DFMA $M := (Q, \Sigma_I, \Sigma_O, \delta, \theta, q^0)$ with $I \subseteq L_I(M)$ and $L_O(M) \subseteq O$ that realizes $\mathcal{D}_I(\cdot)$. Let $[s]$ denote the equivalence class under the equivalence relation $\simeq$ containing the string s. Define a DMA $M := (Q, \Sigma_I, \Sigma_O, \delta, \theta, q^0)$, where (i) $Q := \{[s] \mid s \in$

$\Sigma_I^*\}$, (ii) $q^0 := [\epsilon]$, (iii) for each $s \in \Sigma_I^*$ and $\sigma \in \Sigma_I$ define $\delta([s], \sigma) := [s\sigma]$, and (iv) for each $s \in \Sigma_I^*$ and $\sigma \in \Sigma_I$ define

$$\theta([s], \sigma) := \begin{cases} \mathcal{D}_I(s'\sigma)_{(|\mathcal{D}_I(s')|)} & \text{if exists } s' \in [s] \text{ s.t. } s'\sigma \in I \\ \epsilon & \text{otherwise} \end{cases}$$

Note that the transition function of M is well defined, as $s', s'' \in [s]$ implies $s'\sigma, s''\sigma \in [s\sigma]$ for each $\sigma \in \Sigma_I$. Also note that in the definition of the output function if $s'\sigma \in I$, then $s' \in I$ since I is closed. Hence $\mathcal{D}_I(s')$ and $\mathcal{D}_I(s'\sigma)$ are both defined. Also, since $\mathcal{D}_I(\cdot)$ is causal, $\mathcal{D}_I(s'\sigma)_{(|\mathcal{D}_I(s')|)}$ is defined. Furthermore, it follows from the definition of the equivalence class $[s]$ that if $s', s'' \in [s]$, then $\mathcal{D}_I(s'\sigma)_{(|\mathcal{D}_I(s')|)} = \mathcal{D}_I(s''\sigma)_{(|\mathcal{D}_I(s'')|)}$. Hence the output function is well defined. Moreover, regularity of $\mathcal{D}_I(\cdot)$ implies that $|Q| < \infty$. We claim that M is the desired DFMA. Note that from construction of M we have $L_I(M) = \Sigma_I^*$; hence $I \subseteq L_I(M)$. Also, $L_O(M) = \{s \in O \mid \exists t \in I \text{ s.t. } \mathcal{D}_I(t) = s\}$, which implies that $L_O(M) \subseteq O$. It follows from the definition of the output function of M that for any $s \in I$ we have

$$\theta^*(q^0, s) = \mathcal{D}_I(s^{(1)})\mathcal{D}_I(s^{(2)})_{(|\mathcal{D}_I(s^{(1)})|)} \ldots \mathcal{D}_I(s^{(k)})_{(|\mathcal{D}_I(s^{(k-1)})|)} \ldots \mathcal{D}_I(s^{(|s|)})_{(|\mathcal{D}_I(s^{(|s|-1)})|)} = \mathcal{D}_I(s),$$

where the last equality follows from the fact that $\mathcal{D}_I(\cdot)$ is causal. Thus M realizes the input-output map $\mathcal{D}_I(\cdot)$.

($\Leftarrow$) Assume next that a DFMA $M := (Q, \Sigma_I, \Sigma_O, \delta, \theta, q^0)$ with $I \subseteq L_I(M)$ and $L_O(M) \subseteq O$ is given. We need to show that it realizes a regular causal map $\mathcal{D}_I : I \to O$. For any string $s \in I$, define $\mathcal{D}_I(s) := \theta^*(q^0, s)$. Since $\theta^*(q^0, s)$ is defined for each $s \in L_I(M)$ and $I \subseteq L_I(M)$, it follows that $\mathcal{D}_I(s)$ is defined for each $s \in I$. Also, since $\theta^*(q^0, s) \in L_O(M)$ and $L_O(M) \subseteq O$, it follows that $\mathcal{D}_I(s) = \theta^*(q^0, s) \in O$. Thus M realizes an input-output map. It remains to show that this map is also regular causal. Causality follows from the fact that for $s, t \in I, s \leq t$ implies $\theta^*(q^0, s) \leq \theta^*(q^0, t)$. Regularity follows from the fact that for each $s, s' \in I$, $s \simeq s'$ whenever $\delta^*(q^0, s) = \delta^*(q^0, s')$, since for any $t \in \Sigma_I^*$, $st \in I$ if and only if $s't \in I$, and $st, s't \in I$ implies $\theta^*(q^0, st)_{(|\theta^*(q^0,s)|)} = \theta^*(q^0, s't)_{(|\theta^*(q^0,s')|)}$. On the other hand, if $s, s' \notin I$, then $s \simeq s'$. This proves that the index of the equivalence relation $\simeq$ is no greater than $|Q| + 1 < \infty$. ■

Theorem 3 provides an alternate characterization of causal as well as regular causal maps. We next present a condition for the stability of such maps. Recall that given a DMA (respectively, DFMA) M, the notation G_M is used to denote its equivalent DA (respectively, DFA) representation.

Proposition 4 An input-output map $\mathcal{D}_I : I \to O$ is stable and causal (respectively, regular causal) if and only if it is realizable as a DMA (respectively, DFMA) $M := (Q, \Sigma_I, \Sigma_O, \delta, \theta, q^0)$ with $I \subseteq L_I(M), L_O(M) \subseteq O$ and $\{s_{IO} \in L(G_M) \mid \pi_I(s_{IO}) \in I\} \subseteq L(G_B)$.

Proof: We only prove the second claim, as the first claim can be proved analogously. Again, we only prove the forward implication; the reverse implication can be proved analogously. Assume that $\mathcal{D}_I : I \to O$ is stable and regular causal. Then it follows from Theorem 3 that there exists a DFMA $M := (Q, \Sigma_I, \Sigma_O, \delta, \theta, q^0)$ with $I \subseteq L_I(M)$ and $L_O(M) \subseteq O$

that realizes $\mathcal{D}_I(\cdot)$. We need to show that if $s_{IO} \in L(G_M)$ is such that $\pi_I(s_{IO}) \in I$, then $s_{IO} \in L(G_B)$, i.e., for each $k \leq |s_{IO}|$ the following holds: $\vec{0} \leq [\overrightarrow{\pi_I(s_{IO}^{(k)})} - \overrightarrow{\pi_O(s_{IO}^{(k)})}] \leq \overrightarrow{|B|}$. Fix $k \leq |s_{IO}|$. Since $\pi_I(s_{IO}) \in I$ and I is closed, $s := \pi_I(s_{IO}^{(k)}) \in I$. Also, since $s_{IO} \in L(G_M)$ and $L(G_M)$ is closed, $s_{IO}^{(k)} \in L(G_M)$. Hence $t := \pi_O(s_{IO}^{(k)}) \in L_O(M) \subseteq O$. Note that from the construction of the DFA G_M, we have $t = \theta^*(q^0, s)$, and from the construction of the DMA M we have, $\theta^*(q^0, s) = \mathcal{D}_I(s)$. Hence $t = \mathcal{D}_I(s)$. Since $\mathcal{D}_I(\cdot)$ is stable over I we have: $\vec{0} \leq [\vec{s} - \overrightarrow{\mathcal{D}(s)}] \leq \overrightarrow{|B|}$. This implies that $\vec{0} \leq [\overrightarrow{\pi_I(s_{IO}^{(k)})} - \overrightarrow{\pi_O(s_{IO}^{(k)})}] \leq \overrightarrow{|B|}$. Since $k \leq |s_{IO}|$ is arbitrary, the desired result is obtained. ■

Finally we provide a test for determining whether a given dispatching unit is dispatchable. Recall that the output-composed DA G_{BO} defined in the previous subsection has the property that (i) $\pi_O(L(G_{BO})) \subseteq O$, and (ii) $L(G_{BO}) \subseteq L(G_B)$, which implies that $\{s_{IO} \in L(G_{BO}) \mid \pi_I(s_{IO}) \in I\} \subseteq L(G_{BO}) \subseteq L(G_B)$. Furthermore, if $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable, then $I \subseteq \pi_I(L(G_{BO}))$. Thus if $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable, then it follows from Proposition 4 that the DA G_{BO} can be used to realize a stable and causal input-output map provided it can be represented as a DMA. $G_{BO} := (Q_B \times Q_O, \Sigma_B, \delta_{BO}, (\vec{0}, q_O^0))$ can be represented as a DMA whenever it satisfies the following condition:

$$\textbf{C1}: \qquad \pi_I(G_{BO}) \text{ is deterministic.}$$

In other words, for each admissible buffer state and output DA state, there is *at most one* departure sequence for an arrival event.

Example 9 Consider for example the DFA G_{BO} shown in Figure 4(a). Then it can be seen that at state $(\vec{0}, 3)$, there are transitions with labels (a, ϵ) as well as (a, a). Thus upon arrival of a at this state, it is possible to either buffer a (i.e., output nothing) or output a without violating either the buffer capacity constraint or the output language constraint. Hence the departure sequence for the arrival event a at state $(\vec{0}, 3)$ is not unique, i.e., condition C1 above does not hold in this case. This can also be verified by noting that the corresponding $\pi_I(G_{BO})$ of Figure 4(b) is nondeterministic. ■

If C1 holds, then letting $s_{(\vec{b}, q_O, \sigma)} \in \Sigma_O^{\leq ||B||}$ denote the unique departure sequence for an arrival event σ at a buffer state $\vec{b}$ and output DA state q_O we can represent G_{BO} as a DMA $M_{BO} := (Q_B \times Q_O, \Sigma_I, \Sigma_O, \hat{\delta}_{BO}, \hat{\theta}_{BO}, (\vec{0}, q_O^0))$, where for each $\vec{b} \in Q_B, q_O \in Q_O, \sigma \in \Sigma_I$:

$$\hat{\delta}_{BO}((\vec{b}, q_O), \sigma) := \delta_{BO}((\vec{b}, q_O), (\sigma, s_{(\vec{b}, q_O, \sigma)})); \quad \hat{\theta}_{BO}((\vec{b}, q_O), \sigma) := s_{(\vec{b}, q_O, \sigma)}.$$

It can be easily verified that $L_I(M_{BO}) = \pi_I(L(G_{BO}))$ and $L_O(M_{BO}) = \pi_O(L(G_{BO}))$. Thus we obtain the following sufficient condition for dispatchability of $(I, O, \overrightarrow{|B|})$:

Proposition 5 Suppose C1 holds, i.e., $\pi_I(G_{BO})$ is deterministic. Then $(I, O, \overrightarrow{|B|})$ is dispatchable if and only if $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable.

Proof: Since dispatchability implies conditional dispatchability, it suffices to show that C1 and conditional dispatchability of $(I, O, \overrightarrow{|B|})$ imply its dispatchability. Since C1 holds, the output-composed DA G_{BO} can be equivalently represented as a DMA M_{BO}. It follows from the construction of M_{BO} that $L_I(M_{BO}) = \pi_I(L(G_{BO})), L_O(M_{BO}) = \pi_O(L(G_{BO})) \subseteq O$ and $L(G_{M_{BO}}) = L(G_{BO}) \subseteq L(G_B)$. Furthermore since $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable, it follows from Theorem 2 that $I \subseteq \pi_I(L(G_{BO}))$. Hence we obtain that $I \subseteq L_I(M_{BO}), L_O(M_{BO}) \subseteq O$ and $L(G_{M_{BO}}) \subseteq L(G_B)$. Hence it follows from Proposition 4 that M_{BO} realizes a stable and causal input-output map, i.e., $(I, O, \overrightarrow{|B|})$ is dispatchable. ∎

The result of Proposition 5 can be strengthened to obtain a necessary and sufficient condition for dispatchability. First consider the *input-composition* of G_I and G_{BO}, denoted $G_{IBO} := (Q_I \times Q_B \times Q_O, \Sigma_B, \delta_{IBO}, (q_I^0, \vec{0}, q_O^0))$, and defined as $(q_{IBO} := (q_I, \vec{b}, q_O) \in Q_I \times Q_B \times Q_O, s_{IO} := (\sigma, s) \in \Sigma_B)$:

$$\delta_{IBO}(q_{IBO}, s_{IO}) := \begin{cases} \big(\delta_I(q_I, \sigma), \delta_{BO}((\vec{b}, q_O), s_{IO})\big) & \text{if } \delta_I(q_I, \sigma), \delta_{BO}((\vec{b}, q_O), s_{IO}) \text{ defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Clearly, $L(G_{IBO}) = \{s \in L(G_B) \mid \pi_I(s) \in I \text{ and } \pi_O(s) \in O\}$. Consequently, $L(G_{IBO}) \subseteq L(G)$, $\pi_I(L(G_{IBO})) \subseteq I$ and $\pi_O(L(G_{IBO})) \subseteq O$.

Remark 5 It follows from the definition of G_{IBO} that $I \subseteq \pi_I(L(G_{BO}))$ if and only if $I = \pi_I(L(G_{IBO}))$. Hence Theorem 2 can be rephrased as: A dispatching policy $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable if and only if $I = \pi_I(L(G_{IBO}))$. Similarly, Proposition 5 can be stated as: If $\pi_I(G_{IBO})$ is deterministic, then $(I, O, \overrightarrow{|B|})$ is dispatchable if and only if it is conditionally dispatchable.

Note that in general $\pi_I(G_{IBO})$ may not be deterministic. However, it is easy to construct a subautomaton $G'_{IBO} \leq G_{IBO}$ such that $\pi_I(G'_{IBO})$ is deterministic. If such a subautomaton G'_{IBO} satisfying $I = \pi_I(L(G'_{IBO}))$ exists, then $(I, O, \overrightarrow{|B|})$ is dispatchable. We show below that the converse is also true. We first define the notion of a "canonical" stable and causal input-output map, which requires that whenever the departure sequence pair for a pair of Nerode equivalent arrival sequences $s, s' \in I$ is Nerode equivalent and yields an identical buffer state, then the "future" departure sequence for any "future" arrival sequence t should be identical for s and s'. Formally,

Definition 9 Given a dispatching unit $(I, O, \overrightarrow{|B|})$ and a stable and causal input-output map $\mathcal{D}_{\mathcal{I}}$, it is called *canonical*, if for each $s, s' \in I$ satisfying $s \equiv_I s'$, $\mathcal{D}(s) \equiv_O \mathcal{D}_I(s')$ and $[\vec{s} - \overrightarrow{\mathcal{D}_I(s)}] = [\vec{s'} - \overrightarrow{\mathcal{D}_I(s')}]$, we have $\mathcal{D}_I(st)_{(|\mathcal{D}_I(s)|)} = \mathcal{D}_I(s't)_{(|\mathcal{D}_I(s')|)}$ for each $t \in \Sigma_I^*$.

The next lemma states that a stable and causal input-output map can always be chosen to be a canonical one.

Lemma 3 Given a dispatching unit $(I, O, \overrightarrow{|B|})$, there exists a stable and causal input-output map over I if and only if there exists a canonical such map.

Proof: It suffices to show the necessity. Suppose a stable and causal input-output map $\mathcal{D}_I : I \to O$ is given. If it is not canonical, then there exist $s, s' \in I$ and $t \in \Sigma_I^*$ such that $s \equiv_I s'$, $\mathcal{D}(s) \equiv_O \mathcal{D}_I(s')$, $[\vec{s} - \overrightarrow{\mathcal{D}_I(s)}] = [\vec{s'} - \overrightarrow{\mathcal{D}_I(s')}]$, but $\mathcal{D}_I(st)_{(|\mathcal{D}_I(s)|)} \neq \mathcal{D}_I(s't)_{(|\mathcal{D}_I(s')|)}$. A canonical stable and causal input-output map $\mathcal{D}'_I$ can be obtained from $\mathcal{D}_I$ by defining it to be the same as $\mathcal{D}_I$ except that for every prefix $\hat{t} \leq t$, it maps the arrival sequence $s'\hat{t}$ to the departure sequence $\mathcal{D}_I(s')\mathcal{D}_I(s\hat{t})_{(|\mathcal{D}_I(s)|)}$. By definition, $\mathcal{D}_I(s\hat{t}) = \mathcal{D}_I(s)\mathcal{D}_I(s\hat{t})_{(|\mathcal{D}_I(s)|)} \in O$, so $\mathcal{D}_I(s) \equiv_O \mathcal{D}_I(s')$ implies $\mathcal{D}'_I(s'\hat{t}) = \mathcal{D}_I(s')\mathcal{D}_I(s\hat{t})_{(|\mathcal{D}_I(s)|)} \in O$. Also since $\mathcal{D}_I$ is causal, it follows that $\mathcal{D}'_I$ is causal. Finally since $[\vec{s} - \overrightarrow{\mathcal{D}_I(s)}] = [\vec{s'} - \overrightarrow{\mathcal{D}_I(s')}]$, it follows that

$$\begin{aligned} [\overrightarrow{s\hat{t}} - \overrightarrow{\mathcal{D}_I(s\hat{t})}] &= [\vec{s} - \overrightarrow{\mathcal{D}_I(s)}] + [\vec{\hat{t}} - \overrightarrow{\mathcal{D}_I(s\hat{t})_{(|\mathcal{D}_I(s)|)}}] \\ &= [\vec{s'} - \overrightarrow{\mathcal{D}_I(s')}] + [\vec{\hat{t}} - \overrightarrow{\mathcal{D}'_I(s'\hat{t})_{(|\mathcal{D}_I(s')|)}}] \\ &= [\overrightarrow{s'\hat{t}} - \overrightarrow{\mathcal{D}'_I(s'\hat{t})}], \end{aligned}$$

i.e., the buffer capacity constraint is also satisfied, which implies $\mathcal{D}'_I$ is stable. ■

Theorem 4 A dispatching unit $(I, O, \overrightarrow{|B|})$ is dispatchable if and only if there exists a subautomaton $G'_{IBO} \leq G_{IBO}$ such that $\pi_I(G'_{IBO})$ is deterministic and $I = \pi_I(L(G'_{IBO}))$.

Proof: $(\Rightarrow)$ First assume that $(I, O, \overrightarrow{|B|})$ is dispatchable. We need to show that there exists a subautomaton $G'_{IBO} \leq G_{IBO}$ such that $\pi_I(G'_{IBO})$ is deterministic and $I = \pi_I(L(G'_{IBO}))$. From hypothesis there exists a stable and causal input-output map $\mathcal{D}_I : I \to O$. From Lemma 3 it can be chosen to be canonical. Using this input-output map construct a subautomaton $G'_{IBO} := (Q_I \times Q_B \times Q_O, \Sigma_B, \delta'_{IBO}, (q_I^0, \vec{0}, q_O^0)) \leq G_{IBO}$, where the transition function is defined as ($q = (q_I, \vec{b}, q_O) \in Q_I \times Q_B \times Q_O, s_{IO} = (\sigma_I, s_O) \in \Sigma_I \times \Sigma_O^{\leq \|B\|}$):

$$\delta'_{IBO}(q, s_{IO}) := \begin{cases} \delta_{IBO}(q, s_{IO}) & \exists s_I \in I \text{ s.t. } \delta^*_{IBO}((q_I^0, \vec{0}, q_O^0), (s_I, \mathcal{D}_I(s_I)) = q, \\ & \text{and } \mathcal{D}_I(s_I \sigma_I)_{(|\mathcal{D}(s_I)|)} = s_O \\ \text{undefined} & \text{otherwise} \end{cases}$$

Note if there exists another arrival sequence $s'_I \in I$ such that $\delta^*_{IBO}((q_I^0, \vec{0}, q_O^0), (s'_I, \mathcal{D}_I(s'_I)) = q$, then $s_I \equiv_I s'_I$, $\mathcal{D}_I(s_I) \equiv_O \mathcal{D}_I(s'_I)$, and $[\vec{s_I} - \overrightarrow{\mathcal{D}_I(s_I)}] = [\vec{s'_I} - \overrightarrow{\mathcal{D}_I(s'_I)}] = \vec{b}$. Since $\mathcal{D}_I$ is canonical, this implies $\mathcal{D}(s_I\sigma_I)_{(|\mathcal{D}_I(s_I)|)} = s_O = \mathcal{D}(s'_I\sigma_I)_{(|\mathcal{D}(s'_I)|)}$, i.e., there is at most one departure sequence for the arrival event σ_I. So $\pi_I(G'_{IBO})$ is deterministic.

Since $\pi_I(L(G'_{IBO})) \subseteq \pi_I(L(G_{IBO})) \subseteq I$, it remains to show that the reverse inequality also holds. We use induction on the length of strings in I to prove that if $s_I \in I$, then $s_I \in \pi_I(L(G'_{IBO}))$. In fact we prove a stronger claim:

$$s_I \in I \Rightarrow \delta'^*_{IBO}\left((q_I^0, \vec{0}, q_O^0), (s_I, \mathcal{D}_I(s_I)\right) = \left(\delta_I^*(q_I^0, s_I), [\vec{s_I} - \overrightarrow{\mathcal{D}_I(s_I)}], \delta_O^*(q_O^0, \mathcal{D}_I(s_I))\right). \quad (1)$$

Note that the condition of (1) implies that $(s_I, \mathcal{D}_I(s_I)) \in L(G'_{IBO})$, which in turn implies that $s_I \in \pi_I(L(G'_{IBO}))$. The condition of (1) certainly holds for the zero length string $\epsilon \in I$ since $\mathcal{D}_I(\epsilon) = \epsilon$. Hence the base step holds. In order to prove the induction step, consider $s_I \in I$ and $\sigma_I \in \Sigma_I$ such that $s_I\sigma_I \in I$. Then from induction hypothesis, (1) holds. Let $q_I := \delta_I^*(q_I^0, s_I)$, $\vec{b} := [\vec{s_I} - \overrightarrow{\mathcal{D}_I(s_I)}]$ and $q_O := \delta_O^*(q_O^0, \mathcal{D}_I(s_I))$. Then it follows from the definition of the transition function of G'_{IBO} that

$$[\delta'_{IBO}((q_I, \vec{b}, q_O), (\sigma_I, s_O)) = \delta_{IBO}((q_I, \vec{b}, q_O), (\sigma_I, s_O))] \Longleftrightarrow [s_O = \mathcal{D}_I(s_I\sigma_I)_{(|\mathcal{D}_I(s_I)|)}]. \quad (2)$$

Thus by combining (1) and (2) we obtain the desired result of induction step:

$$\delta'^*_{IBO}\left((q_I^0, \vec{0}, q_O^0), (s_I\sigma_I, \mathcal{D}_I(s_I\sigma_I))\right) = \left(\delta_I^*(q_I^0, s_I\sigma_I), [\overrightarrow{s_I\sigma_I} - \overrightarrow{\mathcal{D}_I(s_I\sigma_I)}], \delta_O^*(q_O^0, \mathcal{D}_I(s_I\sigma_I))\right).$$

($\Leftarrow$) Next assume that there exists a subautomaton $G'_{IBO} \le G_{IBO}$ such that $\pi_I(G'_{IBO})$ is deterministic and $I = \pi_I(L(G'_{IBO}))$. We need to show that $(I, O, \overrightarrow{|B|})$ is dispatchable. Construct an equivalent DMA, M'_{IBO} for the DA G'_{IBO}. This is possible since $\pi_I(G'_{IBO})$ is deterministic. Then $I = \pi_I(L(G'_{IBO})) = L_I(M'_{IBO}), \pi_O(L(G'_{IBO})) = L_O(M'_{IBO}) \subseteq O$ and $L(G'_{IBO}) = L(G_{M'_{IBO}}) \subseteq L(G_B)$. Hence it follows from Proposition 4 that $(I, O, \overrightarrow{|B|})$ is dispatchable. ■

Example 10 Consider the setting of Example 7. As mentioned above, the corresponding DFA G_{BO} is shown in Figure 4(a). Also, as noted in Example 9 condition C1 does not hold in this case. Thus although the triple $(I, O, \overrightarrow{|B|})$ is conditionally dispatchable the test for sufficiency of dispatchability as given in Proposition 5 is not applicable. So we construct the DFA G_{IBO} as shown in Figure 5. Clearly, $\pi_I(G_{IBO})$ is nondeterministic. However, the subautomaton $G'_{IBO} \le G_{IBO}$ lying within the dashed rectangular area of Figure 5 is such that $\pi_I(G'_{IBO})$ is deterministic and $I = \pi_I(L(G'_{IBO}))$. Thus it follows from Theorem 4 that the triple $(I, O, \overrightarrow{B})$ is dispatchable. The required dispatching policy is obtained as discussed in Example 8. ■

7 Conclusion

Quantitative design of dispatching policies, that involves queueing theory based analysis, has been widely studied. In contrast, this paper deals with *qualitative* design of dispatching policies and addresses the problem of whether a given input-output discrete event system can be realized as a stable dispatching unit, i.e., a system consisting of a set of finite capacity buffers. We have formalized the notion of a dispatching unit as a triple consisting of an input language constraint (describing the sequences in which parts may arrive at the dispatching unit), an output language constraint (describing sequences in which parts may be dispatched), and a buffer capacity constraint. Dispatching policies are introduced as

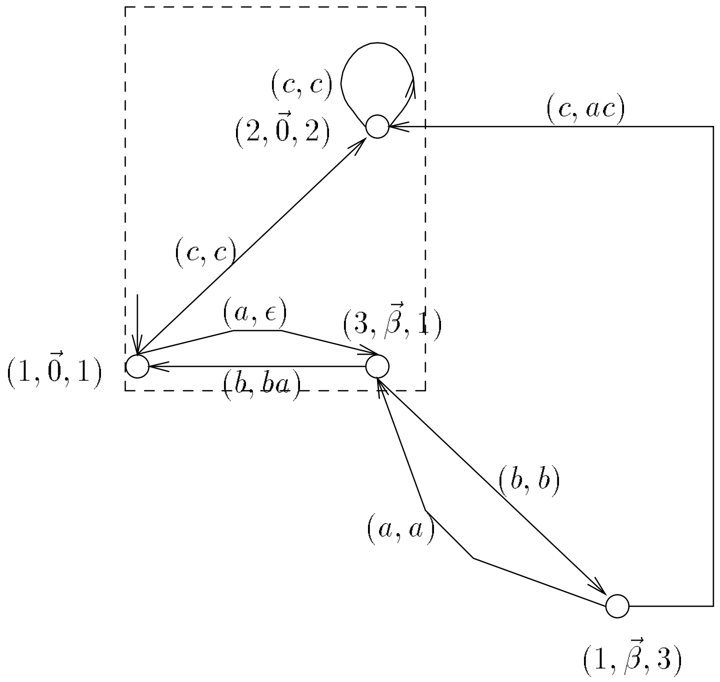


Figure 5: Diagram illustrating G_{IBO} and G'_{IBO}

stable and causal input-out maps, and notions of conditionally stable, stable, conditionally dispatchable, and dispatchable dispatching units is introduced and conditions for their existence, and techniques for synthesizing desired input-output maps whenever they exist are presented.

There are a number of interesting open problems for future research: (i) algebraic properties such as closure under union and intersection of dispatching units of each type needs to be studied; thus if a certain dispatching policy does not exist for a given dispatching unit, then design of *minimally restrictive* [26] dispatching policies can be considered; (ii) the assumption that the buffer capacities are given may be relaxed, and design of dispatching policies with respect to any finite buffer capacity vector may be considered; (iii) alternative techniques for determining dispatching policies such as on-line techniques as studied in [8, 4], and integer-programming based techniques for analyzing vector discrete event systems as reported in [18], should be developed for facilitating computationally efficient design; (iv) stability properties of a *network* of dispatching units ought to be investigated so that a "modular" notion of stability of I/O DESs can be developed.

Acknowledgment

We thank the anonymous reviewers for their valuable comments.

References

[1] S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. F. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, 38(7):1040–1059, July 1993.

[2] Y. Brave and M. Heymann. On stabilization of discrete event processes. *International Journal of Control*, 51(5):1101–1117, 1990.

[3] C. Chase and P. J. Ramadge. On real time scheduling policies for flexible manufacturing systems. *IEEE Transactions on Automatic Control*, 37(4):491–496, April 1992.

[4] S. L. Chung, S. Lafortune, and F. Lin. Limited lookahead policies in supervisory control of discrete event systems. *IEEE Transactions of Automatic Control*, 37(12):1921–1935, December 1992.

[5] Y. Dallery, R. David, and X. Xie. Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on Automatic Control*, 9(34):943–953, September 1989.

[6] S. B. Gershwin. Hierarchical flow control: A framework for scheduling and planning discrete events in manufacturing systems. *Proceedings of IEEE*, 77(1):195–209, January 1989.

[7] S. Ginsburg. *Mathematical Theory of Context-Free Languages*. Addison Wesley, Reading, MA, 1966.

[8] M. Heymann and F. Lin. On-line control of partially observed discrete event systems. *Discrete Event Dynamical Systems: Theory and Applications*, 4(3):221–236, July 1994.

[9] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1985.

[10] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA, 1979.

[11] R. E. Kalman, P. L. Falb, and M. A. Arbib. *Topics in Mathematical System Theory*. McGraw-Hill, Inc., 1969.

[12] V. S. Kouikoglou and Y. A. Phillis. An exact discrete-event model and control policies for production lines with buffers. *IEEE Transactions on Automatic Control*, 36(5):515–527, May 1991.

[13] P. R. Kumar and T. I. Seidman. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control*, 35(3):289–298, March 1990.

[14] R. Kumar and V. K. Garg. *Modeling and Control of Logical Discrete Event Systems.* Kluwer Academic Publishers, Boston, MA, 1995.

[15] R. Kumar, V. K. Garg, and S. I. Marcus. Language stability and stabilizability of discrete event dynamical systems. *SIAM Journal of Control and Optimization*, 31(5):1294–1320, September 1993.

[16] R. Kumar, V. K. Garg, and S. I. Marcus. Logical design of a dispatching unit. In *Proceedings of 1993 American Control Conference*, pages 1198–1202, June 1993.

[17] Y. Li and W. M. Wonham. Control of vector discrete event systems I - the base model. *IEEE Transactions on Automatic Control*, 38(8):1214–1227, August 1993.

[18] Y. Li and W. M. Wonham. Control of vector discrete event systems II - controller synthesis. *IEEE Transactions on Automatic Control*, 39(3):512–531, 1994.

[19] S. H. Lu and P. R. Kumar. Distributed scheduling based on due dates and buffer priorities. *IEEE Transactions on Automatic Control*, 36(12):1406–1416, December 1991.

[20] N. Lynch. I/O automata: a model for discrete event systems. Technical report, Massachusetts Institute of Technology, Cambridge, MA, 1988.

[21] C. M. Ozveren and A. S. Willsky. Output stabilizability of discrete event dynamical systems. *IEEE Transactions on Automatic Control*, 36(8):925–935, 1991.

[22] C. M. Ozveren, A. S. Willsky, and P. J. Antsaklis. Stability and stabilizability of discrete event dynamical systems. *Journal of ACM*, 38(3):730–752, July 1991.

[23] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, Reading, MA, 1987.

[24] R. J. Parikh. On context-free languages. *Journal of the Association for Computing Machinery*, 13:570–581, 1966.

[25] J. R. Perkins and P. R. Kumar. Stable, distributed, real time scheduling of flexible manufacturing/assembly/disassembly systems. *IEEE Transactions of Automatic Control*, 34(2):139–148, February 1989.

[26] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[27] D. Towsley, P. D. Sparaggis, and C. G. Cassandras. Optimal routing and buffer allocation for a class of finite capacity queueing systems. *IEEE Transactions on Automatic Control*, 37(9):1446–1451, September 1992.