Synchronous Elastic Networks

Sava Krstić
Strategic CAD Labs, Intel Corporation
Hillsboro, Oregon, USA

Jordi Cortadella
Universitat Politècnica de Catalunya
Barcelona, Spain

Mike Kishinevsky, John O’Leary
Strategic CAD Labs, Intel Corporation
Hillsboro, Oregon, USA

Abstract—We formally define—at the stream transformer
level—a class of synchronous circuits that tolerate any variabil-
ity in the latency of their environment. We study behavioral
properties of networks of such circuits and prove fundamental
compositionality results. The paper contributes to bridging
the gap between the theory of latency-insensitive systems and
the correct implementation of efficient control structures for them.

I. INTRODUCTION

The conventional abstract model for a synchronous circuit is
a machine that reads inputs and writes outputs at every cycle.
The outputs at cycle \(i\) are produced according to a calculation
that depends on the inputs at cycles \(0, \ldots, i\). Computations
and data transfers are assumed to take zero delay.

Latency-insensitive design by Carloni et al. [2] aims to relax
this model by elasticizing the time dimension and so decou-
pling the cycles from the calculations of the circuit. It enables
the design of circuits tolerant to any discrete variation (in
the number of cycles) of the computation and communication
delays. With this modular approach, the functionality of the
system only depends on the functionality of its components
and not on their timing characteristics.

The motivation for latency-insensitive design comes from
the difficulties with timing and communication in nanoscale
technologies. The number of cycles required to transmit data
from a sender to a receiver is governed by the distance
between them, and often cannot be accurately known until
the chip layout is generated late in the design process. Conven-
tional design approaches require fixing the communication
latencies up front, and these are difficult to amend when
layout information finally becomes available. Elastic circuits
offer a solution to this problem. In addition, their modularity
promises novel methods for microarchitectural design that
can use variable-latency components and tolerate static and
dynamic changes in communication latencies, while—unlike
asynchronous circuits—still employing standard synchronous
design tools and methods.

Cortadella et al. [4] present a simple elastic protocol, called
SELF (Synchronous Elastic Flow) and describe methods for
efficient implementation of elastic systems and for conversion
of regular synchronous designs into elastic form. Inspired by
the original work on latency-insensitive design [2], SELF also
differs from it in ways that render the theory developed in [2]
hardly applicable.

In this paper we give theoretical foundations of SELF: a
novel and arguably more practicable definition of elasticity,
and the basic compositionality results. For space reasons, the
proofs are omitted, but are available in the technical report
[7].

A. Overview

Figure 1(a) depicts the timing behavior of a conventional
synchronous adder that reads input and produces output data
at every cycle (boxes represent cycles). In this adder, the \(i\)-th
output value is produced at the \(i\)-th cycle. Figure 1(b) depicts
a related behavior of an elastic adder—a synchronous circuit
too—in which data transfer occurs in some cycles and not in
others. We refer to the transferred data items as tokens and we
say that idle cycles contain bubbles.

Put succinctly, elasticization decouples cycle count from
token count. In a conventional synchronous circuit, the \(i\)-th
token of a wire is transmitted at the \(i\)-th cycle, whereas in
a synchronous elastic circuit the \(i\)-th token is transmitted at
some cycle \(k \geq i\).

Turning a conventional synchronous adder into a syn-
chronous elastic adder requires a communication discipline
that differentiates idle from non-idle cycles (bubbles from
tokens). In SELF, this is implemented by a pair of single-
bit control wires: \textit{Valid} and \textit{Stop}. Every input or output wire
\(Z\) in a synchronous component is associated to a channel
in the elastic version of the same component. The channel is a
triple of wires \((Z, \text{valid}_Z, \text{stop}_Z)\), with \(Z\) carrying the data and
the other two wires implementing the control bits, as shown
in Figure 2(b). A token is transferred on this channel when
\(\text{valid}_Z \land \neg \text{stop}_Z\): the sender sends valid data and the receiver
is ready to accept it; see Figure 1(b). Additional constraints that
ensure correct elastic behavior are given in Section III.
There we define precisely the class of elastic circuits and what
it means for a circuit \(A^e\) to be an elastization of a given circuit
\(A\). In particular, our definition implies liveness: \(A^e\) produces
infinite streams of tokens if its environment produces infinite
streams of tokens at the input channels and is ready to accept
infinite streams at the output channels.

Suppose \(\mathcal{N}\) is a network of standard (non-elastic) compo-
nents, as in Figure 2(a). Suppose we then take elasticizations of

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig1.png}
\caption{(a) Conventional synchronous adder, (b) Synchronous elastic adder.}
\end{figure}
these standard components and join their channels accordingly, as in Figure 2(b), ignoring the buffer. Will the resulting network $\mathcal{N}^e$ be an elasticization of $\mathcal{N}$? Will it be elastic at all? These fundamental questions are answered by Theorem 4 of Section IV, which is the main result of the paper. The answers are “yes”, provided a certain graph $\Delta^e(\mathcal{N}^e)$ associated with $\mathcal{N}^e$ is acyclic. This graph captures the information about paths inside elastic systems that contain no tokens—analogous to combinational paths in ordinary systems. Importantly, $\Delta^e(\mathcal{N}^e)$ can be constructed using only local information (the “sequentiality interfaces”) of the individual elastic components.

Since elastic networks tolerate any variability in the latency of the components, empty FIFO buffers can be inserted in any channel, as shown in Figure 2(b), without changing the functional behavior of the network. This practically important fact is proved as a consequence of Theorem 4.

Synchronous circuits are modeled in this paper as stream transformers, called machines. This well-known technique (see [8] and references therein) appears to be quite underdeveloped. Our rather lengthy preliminary Section II elaborates the necessary theory of networks of machines, culminating with a surprisingly novel combinational loop theorem (Theorem 1).

Figure 3 illustrates Theorem 1 and, by analogy, Theorem 4 as well. It relies on the formalization of the notion of combinational dependence at the level of input-output wire pairs. Each input-output pair of a machine is either sequential or not, and the set of sequential pairs provides a machine’s “sequentiality interface”. When several machines are put together into a network $\mathcal{N}$, their sequentiality interfaces define the graph $\Delta(\mathcal{N})$, the acyclicity of which is a test for the network to be a legitimate machine itself.

Elasticizations of ordinary circuits are not uniquely defined. On the other hand, for every elastic machine $A$ there is a unique standard machine, denoted $A^s$, that corresponds to it. We do not discuss any specific elasticization procedures in this paper, but state our results in the form that only involves elastic machines and their unique standard counterparts. This makes the results applicable to multiple elasticization procedures.

B. Related Work

Carloni et al. [2] pioneered a theory of latency-insensitive circuits based on their notion of patient processes. Patient processes are defined at a high level of abstraction that models communication on a channel only by “token or bubble”, leaving implementation protocol(s) unspecified. In the companion paper [3], Carloni et al. give an incomplete description of an implementation protocol. Assuming our recovery of that protocol (let us call it LID) is accurate, its transfer condition is more complex than that of SELF (Figure 4) and consequently LID requires significantly more complex implementation. For example, conversion of a regular design into LID form needs a wrapper or registers around every module, increasing the latency of each module’s computation by two cycles—a penalty that is not required in the SELF elasticization. There might also be practical challenges in interfacing a LID system with an existing non-LID module, requiring the latter to generate stop signals with complex semantics.

We emphasize that the limitations of LID implementations are not inherent to the concept of patient processes. Regarding latency properties, they do not seem to be more limited than elastic systems. Still, it turns out that patient processes are not general enough to model elastic systems as we define them in Section III. This we prove in Section V where patient processes and elastic systems are compared as alternative formalizations of latency-insensitive circuits.

Suhaib et al. [12] revisited and generalized Carloni’s elasticization procedure, validating its correctness by a simulation method based on model checking.

Lee et al. [9] study causality interfaces (pairwise input-output dependencies) and are “interested in existence and uniqueness of the behavior of feedback composition”, but do not go as far as deriving a combinational loop theorem.

In their work on design of interlock pipelines [6], Jacobson et al. use a protocol equivalent to SELF, without explicitly
specifying it.

Manohar and Martin discuss “slack elasticity” of asynchronous implementations in [10]. Their slack elasticity conditions relate to the structure of choices in the asynchronous specification. Unlike [10], in the current paper we deal with synchronous systems and we take a black box view of their control—no information about the control flow (and hence on the structure of choices) is ever used. Instead the connectivity information corresponding to the system data-flow is used for elasticization. Conservatively ignoring control flow may lead to a performance penalty, but simplifies the translation to an elastic system.

II. CIRCUITS AS STREAM FUNCTIONS

In this section we introduce machines as a mathematical abstraction of circuits without combinational cycles. For simplicity, this abstraction implicitly assumes that all sequential elements inside the circuit are initialized. Extending to partially initialized systems appears to be trivial. While there is a large body of work studying circuits or equivalent objects with good (e.g. constructive [11]) combinational cycles and their composition (e.g. [5]), we deliberately restrict consideration to the fully acyclic objects, since neither logic synthesis nor timing analysis can properly treat circuits with combinational cycles.

Most of the effort in this section goes into establishing modularity conditions guaranteeing that a system obtained as a mathematical description of a circuit is a machine itself.

A. Streams

A stream over \( A \) is an infinite sequence whose elements belong to the set \( A \). The first element of a stream \( a \) is referred to by \( a[0] \), the second by \( a[1] \), etc. For example, the equation \( a[i] = 3i + 1 \) describes the stream \( 1, 4, 7, \ldots \).

The set of all streams will be denoted \( A^\infty \). Occasionally we will need to consider finite sequences too; the set of all, finite or infinite, sequences over \( A \) is denoted \( A^* \).

We will write \( a \sim_k b \) to indicate that the streams \( a \) and \( b \) have a common prefix of length \( k \). The equivalence relations \( \sim_0, \sim_1, \sim_2, \ldots \) are progressively finer and have trivial intersection. Thus, to prove two sequences \( a \) and \( b \) are equal, it suffices to show \( a \sim_k b \) holds for every \( k \). Note also that \( a \sim_0 b \) holds for every \( a \) and \( b \).

We will use the equivalence relations \( \sim_k \) to express properties of systems and machines viewed as multivariate stream functions. All these properties will be derived from the following two basic properties of single-variable stream functions \( f: A^\infty \rightarrow B^\infty \).

- **causality**: \( \forall a, b \in A^\infty, \forall k \geq 0. a \sim_k b \Rightarrow f(a) \sim_k f(b) \)
- **contraction**: \( \forall a, b \in A^\infty, \forall k \geq 0. a \sim_k b \Rightarrow f(a) \sim_{k+1} f(b) \)

Informally, \( f \) is causal if (for every \( a \)) the first \( k \) elements of \( f(a) \) are determined by the first \( k \) elements of \( a \), and \( f \) is contractive if the first \( k \) elements of \( f(a) \) are determined by the first \( k - 1 \) elements of \( a \).

**Lemma 1**: If \( f: A^\infty \rightarrow A^\infty \) is contractive, then it has a unique fixpoint.

**Remark.** One can define the distance \( d(a,b) \) between sequences \( a \) and \( b \) to be \( 1/2^k \), where \( k \) is the length of the largest common prefix of \( a \) and \( b \). This gives the sets \( A^\infty \) and \( A^\omega \) the structure of complete metric spaces and Lemma 1 is an instance of Banach Fixed Point Theorem. See the review paper [8] for more details and references about the metric semantics of systems and [13] for “diadic arithmetic of circuits”. We choose not to use the metric space terminology in this paper since all “metric reasoning” we need can be as easily done with equivalence relations \( \sim_k \) instead. See [11] for principles of reasoning with such “converging equivalence relations” in more general contexts.

B. Systems

Suppose \( W \) is a set of typed wires; all we know about an individual wire \( w \) is a set \( \text{type}(w) \) associated to it. A \( W \)-behavior is a function \( \sigma \) that associates a stream \( \sigma.w \in \text{type}(w)^\infty \) to each wire \( w \in W \). The set of all \( W \)-behaviors will be denoted \( \llbracket W \rrbracket \). Slightly abusing the notation, we will also write \( \llbracket w \rrbracket \) for the set \( \text{type}(w)^\infty \). Notice that the equivalence relations \( \sim_k \) extend naturally from streams to behaviors:

\[ \sigma \sim_k \sigma' \quad \text{iff} \quad \forall w \in W. \sigma.w \sim_k \sigma'.w \]

Notice also that a \( W \)-behavior \( \sigma \) can be seen as a single stream \( (\sigma[0], \sigma[1], \ldots) \) of \( W \)-states, where a state is an assignment of a value in \( \text{type}(w) \) to each wire \( w \).

**Definition 1**: A \( W \)-system is a subset of \( \llbracket W \rrbracket \).

**Example.** A circuit that at each clock cycle receives an integer as input and returns the sum of all previously received inputs is described by the \( W \)-system \( S \), where \( W \) consists of two wires \( u, v \) of type \( Z \), and \( S \) consists of all stream pairs \( (a, b) \in Z^\infty \times Z^\infty \) such that \( b[0] = 0 \) and \( a[n] = a[0] + \cdots + a[n-1] \) for \( n > 0 \). Each stream pair \( (a, b) \) represents a behavior \( \sigma \) such that \( \sigma.u = a \) and \( \sigma.v = b \).

We will use wires as typed variables in formulas meant to describe system properties. The formulas are built using ordinary mathematical and logical notation, enhanced with temporal operators next, always, and eventually, denoted respectively by \( (\_)^+, G, F \). As an illustration, the system \( S \) in the example above is characterized by the property \( v = 0 \land G(v^+ = v + u) \). Also, one has \( S \models F.G(u > 0) \Rightarrow F.G(v > 1000) \), where \( \models \) is used to denote that a formula is true of a system.

C. Operations on Systems

If \( W' \subseteq W \), there is an obvious projection map \( \sigma \mapsto \sigma \downarrow W' : \llbracket W \rrbracket \rightarrow \llbracket W' \rrbracket \). These projections are all one needs for the definition of the following two basic operations on systems.

**Definition 2**: (a) If \( S \) is a \( W \)-system and \( W' \subseteq W \), then hiding \( W' \) in \( S \) produces a \( (W - W') \)-system \( \text{hide}_{W'}(S) \) defined by

\[ \tau \in \text{hide}_{W'}(S) \quad \text{iff} \quad \exists \sigma \in S. \tau = \sigma \downarrow (W - W'). \]
The composition of a $W_1$-system $S_1$ and a $W_2$-system $S_2$ is a $(W_1 \cup W_2)$-system $S_1 \sqcup S_2$ defined by

$$\sigma \in S_1 \sqcup S_2 \iff \sigma \downarrow W_1 \in S_1 \land \sigma \downarrow W_2 \in S_2.$$ 

If $W$ and $W'$ are disjoint wire sets, $\sigma \in \llbracket W \rrbracket$, and $\tau \in \llbracket W' \rrbracket$, then there is a unique behavior $\sigma \downarrow W = \tau \downarrow W'$ such that $\sigma = \tau \downarrow W$ and $\tau = \tau \downarrow W'$. This “product” of behaviors will be written as $\sigma = \tau \uparrow W$. If $W$ is the empty set, then $\llbracket W \rrbracket$ has one element—a “trivial behavior” that is also a multiplicative unit for the product operation $\uparrow$.) We will also use the notation $[u \mapsto a, v \mapsto b, \ldots]$ for the $\{u, v, \ldots\}$-behavior $\sigma$ such that $\sigma.u = a, \sigma.v = b$, etc.

Hiding and composition suffice to define complex networks of systems. To model identification of wires, we use simple connection systems: by definition, $Conn(u, v)$ is the $\{u, v\}$-system consisting of all behaviors $\sigma$ such that $\sigma.u = \sigma.v$.

Now if $S_1, \ldots, S_m$ are given systems and $u_1, \ldots, u_n, v_1, \ldots, v_n$ are some of their wires, the network obtained from these systems by identifying each wire $u_i$ with the corresponding wire $v_i$ (of equal type) is the system

$$\llbracket S_1, \ldots, S_m \mid u_1 = v_1, \ldots, u_n = v_n \rrbracket$$

defined as hide$_{\{u_1, \ldots, u_n, v_1, \ldots, v_n\}}(S)$, where

$$S = S_1 \sqcup \cdots \sqcup S_m \sqcup Conn(u_1, v_1) \sqcup \cdots \sqcup Conn(u_n, v_n).$$

The simplest case ($m = n = 1$) of networks is the construction

$$\llbracket S \mid u = v \rrbracket = \text{hide}_{\{u, v\}}(S \sqcup Conn(u, v)),$$

used for a feedback definition in Section II-E. A behavior $\sigma$ belongs to $\llbracket S \mid u = v \rrbracket$ if and only if $\sigma \uparrow (u \mapsto a, v \mapsto a) \in S$ for some $a \in \llbracket u \rrbracket$.

D. Machines

Suppose $I$ and $O$ are disjoint sets of wires, called inputs and outputs, correspondingly. By definition, an $(I, O)$-system is just an $(I \sqcup O)$-system. Consider the following properties of an $(I, O)$-system $S$.

**Deterministic:**

$$\forall \omega, \omega' \in S. \ \omega \downarrow I = \omega' \downarrow I \Rightarrow \omega \downarrow O = \omega' \downarrow O$$

**Functional:**

$$\forall \sigma \in \llbracket I \rrbracket, \exists ! \tau \in \llbracket O \rrbracket. \ \sigma \uparrow \tau \in S$$

**Causal:**

$$\forall \omega, \omega' \in S. \ \forall k \geq 0. \ \omega \downarrow I \sim_k \omega' \downarrow I \Rightarrow \omega \downarrow O \sim_k \omega' \downarrow O$$

Clearly, functionality implies determinism. Conversely, a deterministic system is functional if and only if it accepts all inputs. Note also that causality implies determinism: if $\omega \downarrow I = \omega' \downarrow I$, then $\omega \downarrow I \sim_k \omega' \downarrow I$ holds for every $k$, so $\omega \downarrow O \sim_k \omega' \downarrow O$ holds for every $k$ too, so $\omega \downarrow O = \omega' \downarrow O$.

**Definition 3:** An $(I, O)$-machine is an $(I, O)$-system that is both functional and causal.

A functional system $S$ uniquely determines and is determined by the function $F : \llbracket I \rrbracket \rightarrow \llbracket O \rrbracket$ such that $F(\sigma) = \tau$ holds if and only if $\sigma \uparrow \tau \in S$. The causality condition for such $S$ can be also written as follows:

$$\forall \sigma, \sigma' \in \llbracket I \rrbracket. \ \forall k \geq 0. \ \sigma \sim_k \sigma' \Rightarrow F(\sigma) \sim_k F(\sigma').$$

The system in the example in Section II-B is a machine if we regard $u$ as an input wire and $v$ as an output wire. The same is true of the system $Conn(u, v)$: its associated function $F$ is the identity function.

**E. Feedback on Machines**

We will use the term feedback for the system $\llbracket S \mid u = v \rrbracket$ as mentioned in Section II-C when $S$ is a machine and the wires $u$ and $v$ of the same type are an input and output of $S$ respectively. Our concern now is to understand under what conditions the feedback produces a machine.

To fix the notation, assume $S$ is an $(I, O)$-machine given by $F : \llbracket I \rrbracket \rightarrow \llbracket O \rrbracket$, with wires $u \in I, v \in O$ of the same type $A$. By the note at the end of Section II-C, we have that for every $\sigma \in \llbracket I \setminus \{u\} \rrbracket$ and $\tau \in \llbracket O \setminus \{v\} \rrbracket$,

$$\sigma \uparrow \tau \in \llbracket S \mid u = v \rrbracket$$

if and only if

$$\exists a \in A^\infty. \ F(\sigma \uparrow [u \mapsto a]) = \tau \uparrow [v \mapsto a].$$

so $\llbracket S \mid u = v \rrbracket$ is functional when the function $F^{\sigma}_u : A^\infty \rightarrow A^\infty$ defined by $F^{\sigma}_u(a) = F(\sigma \uparrow [u \mapsto a])$ has a unique fixpoint. By Lemma 1, this is guaranteed if $F^{\sigma}_u$ is contractive.

The following definition introduces the key concept of sequentiality that formalizes the intuitive notion that there is no combinational dependence of a given output wire on a given input wire. Sequentiality of the pair $(u, v)$ easily implies contractivity of $F^{\sigma}_u$ for all $\sigma$.

**Definition 4:** The pair $(u, v)$ is sequential for $S$ if for every $\sigma, \sigma' \in \llbracket I \rrbracket$ and every $k \geq 0$

$$\sigma.u \sim_k \sigma'.u \land \forall x \in I \setminus \{u\}. (\sigma.x \sim_k \sigma'.x) \Rightarrow F(\sigma).v \sim_k F(\sigma').v$$

**Lemma 2 (Feedback):** If $(u, v)$ is a sequential input-output pair for a machine $S$, then the feedback system $\llbracket S \mid u = v \rrbracket$ is a machine too.

**Example.** Consider the system $S$ with $I = \{u, v\}$, $O = \{w, z\}$, specified by equations

$$w = u \oplus ((0) \# v) \quad z = v \oplus v,$$

where all wires have type $Z$, the symbol $\oplus$ denotes the componentwise sum of streams, and $\#$ denotes concatenation. Since $z$ does not depend on $u$, the pair $(u, z)$ is sequential. The pair $(v, w)$ is also sequential since to compute a prefix of $w$ it suffices to know (a prefix of the same size of $u$ and) a prefix of smaller size of $v$. The remaining two input-output pairs $(u, w)$ and $(v, z)$ are not sequential.

To find the machine $\llbracket S \mid v = w \rrbracket$, we need to solve the equation $v = u \oplus ((0) \# v)$ for $v$. For each $u = (a_0, a_1, a_2, \ldots)$, the equation has a unique solution $v = \hat{u} = (a_0, a_0 + a_1, a_0 + a_1 + a_2, \ldots)$. Substituting the solution into $z = v \oplus v$, we obtain
Consider a network \( N = \langle S_1, \ldots, S_m | u_1 = v_1, \ldots, u_n = v_n \rangle \), where \( S_1, \ldots, S_m \) are machines with disjoint wire sets and the pairs \( (u_1, v_1), \ldots, (u_n, v_n) \) involve \( n \) distinct input wires \( u_i \) and \( n \) distinct output wires \( v_i \). (There is no assumption that \( u_i, v_i \) belong to the same machine \( S_j \).) Our goal is to understand under what conditions the system \( N \) is a machine.

Note that \( N = \langle S | u_1 = v_2, \ldots, u_n = v_n \rangle \), where \( S = S_1 \cup \cdots \cup S_m \). It is easy to check that an input-output pair \( (u, v) \) of \( S \) is sequential if either (1) \( (u, v) \) is sequential for some \( S_i \), or (2) \( u \) and \( v \) belong to different machines. Thus, the information about sequentiality of input-output pairs of the “parallel composition” machine \( S \) is readily available from the sequentiality information about the component machines \( S_i \), and our problem boils down to determining when a multiple feedback operation performed on a single machine results in a system that is itself a machine.

Simultaneous feedback specified by a set of two or more input-output pairs of a machine does not necessarily produce a machine even if all pairs involved are sequential. Indeed, in the example in Section II-E, we had a system \( S \) with two sequential pairs \( (u, z) \) and \( (v, w) \), but \( (u, z) \) ceases to be sequential for \( \langle S | v = w \rangle \). Indeed, if \( z \) and \( u \) are related by \( z = \hat{u} \oplus \check{u} \), then knowing a prefix of length \( k \) of \( z \) requires knowing the prefix of the same length of \( u \); a shorter one would not suffice.

To ensure that a multiple feedback construction produces a machine, one needs to show that, in addition to the wire pairs to be identified, sufficiently many other input-output pairs are also sequential. A precise formulation for a double feedback is given by a version of the Bekić Lemma: for the system \( \langle S | u = w, v = z \rangle \) to be a machine, it suffices that three pairs of wires be sequential—\( (u, w), (v, z) \), and one of \( (u, z), (v, w) \). This non-trivial auxiliary result is needed for the proof of Theorem 1 below, and is a special case of it.

Given an \((I, O)\)-machine \( S \), let its dependency graph \( \Delta(S) \) have the vertex set \( I \cup O \) and directed edges that go from \( u \) to \( v \) for each pair \((u, v) \in I \times O \) that is not sequential. For a network system \( N = \langle S_1, \ldots, S_m | u_1 = v_1, \ldots, u_n = v_n \rangle \), its graph \( \Delta(N) \) is then defined as the direct sum of graphs \( \Delta(S_1), \ldots, \Delta(S_m) \) with each vertex \( u_i \) (\( 1 \leq i \leq n \)) identified with the corresponding vertex \( v_i \) (Figure 3).

**Theorem 1 (Combinational Loop Theorem):** The network system \( N \) is a machine if the graph \( \Delta(N) \) is acyclic.

### III. Elastic Machines

In this section we give the definition of elastic machines. Its four parts—input-output structure, persistence conditions, liveness conditions, and the transfer determinism condition—are covered by Definitions 5-8 below.

**A. Input-output Structure, Channels, and Transfer**

We assume that the set of wires is partitioned into data, valid, and stop wires, so that for each data wire \( X \) there exist associated wires \( \text{valid}_X \) and \( \text{stop}_X \) of boolean type. (In actual circuit implementations, \text{valid}_X \) and \( \text{stop}_X \) need not be physical wires; it suffices that they be appropriately encoded.)

**Definition 5:** Let \( I, O \) be disjoint sets of data wires. An \([I, O]\)-system is an \((I', O')\)-machine, where \( I' = I \cup \{ \text{valid}_X | X \in I \} \cup \{ \text{stop}_Y | Y \in O \} \) and \( O' = O \cup \{ \text{valid}_X | Y \in O \} \cup \{ \text{stop}_X | X \in I \} \).

The triples \( \langle X, \text{valid}_X, \text{stop}_X \rangle \) (for \( X \in I \)) and \( \langle Y, \text{valid}_Y, \text{stop}_Y \rangle \) (for \( Y \in O \)) are to be thought of as elastic input and output channels of the system.

Let \( \text{transfer}_Z \) be a shorthand for \( \text{valid}_Z \land \neg \text{stop}_Z \) and say that transfer along \( Z \) occurs in a state \( s \) if \( s \models \text{transfer}_Z \).

Given a behavior \( \sigma = (\sigma[0], \sigma[1], \sigma[2], \ldots) \) of an \([I, O]\)-system and \( Z \in I \cup O \), let \( \sigma_Z \) be the sequence (perhaps finite!) obtained from \( \sigma.Z = (\sigma[0].Z, \sigma[1].Z, \sigma[2].Z, \ldots) \) by deleting all entries \( \sigma[i].Z \) such that transfer along \( Z \) does not occur in \( \sigma[i] \).

The transfer behavior \( \sigma^T \) associated with \( \sigma \) is then defined by \( \sigma^T.Z = \sigma_Z \).

If all sequences \( \sigma_Z \) are infinite, then \( \sigma^T \) is an \((I \cup O)\)-behavior; in general, however, we only have \( \sigma_Z \in \text{type}(Z)^\omega \).

For each wire \( Z \) of an \([I, O]\)-system \( S \) we introduce an auxiliary transfer counter variable \( \text{tct}_Z \) of type \( Z \). The counters serve for expressing system properties related to transfer. By definition, \( \text{tct}_Z \) is equal to the number of states that precede the current state and in which transfer along \( Z \) has occurred. That is, for every behavior \( \sigma \) of \( S \), we have \( \sigma.\text{tct}_Z = (t_0, t_1, \ldots) \), where \( t_k \) is the number of indices \( i \) such that \( i < k \) and transfer along \( Z \) occurs in \( \sigma[i] \).

The notation \( \text{min}_{S}.\text{tct}_Z \), for any subset \( S \) of \( I \cup O \) will be used to denote the smallest of the numbers \( \text{tct}_Z \), where \( Z \in S \).

**B. Definition of Elasticity**

An elastic component, when ready to communicate over an output channel must remain ready until the transfer takes place.

**Definition 6:** The persistence conditions for an \([I, O]\)-system \( S \) are given by

\[
S \models G(\text{valid}_Y \land \text{stop}_Y \Rightarrow (\text{valid}_Y)^+ \land Y^+ = Y)
\]  

for every \( Y \in O \).

The conjunct \( Y^+ = Y \) can be removed from (1) without affecting the definition of elastic machines (it follows from other conditions). The most useful consequence of persistence is the “handshake lemma”:

\[
S \models G F \text{valid}_Y \land G F \neg \text{stop}_Y \Rightarrow GF \text{transfer}_Y
\]

Liveness of an elastic component is expressed in terms of token count: if all input channels have seen \( k \) transfers and there is an output channel that has seen less, then the communication on output channels with the minimum amount of transfer must be eventually offered. The following definition formalizes this,
together with a similar commitment to eventual readiness on input channels. (See also Figure 5.)

**Definition 7:** The liveness conditions for an \([I, O]\)-system are given by

\[
S \models G (\min_{\text{tct}_{I,O}} = \text{tct}_Y \land \min_{\text{tct}_I} > \text{tct}_Y \Rightarrow \text{F valid}_Y)(2)
\]

\[
S \models G (\min_{\text{tct}_{I,O}} = \text{tct}_X \Rightarrow \text{F \neg stop}_X)
\]

(3)

for every \(Y \in O\) and every \(X \in I\).

In practice, elastic components will satisfy simpler (but stronger) liveness properties; e.g. remove \(\min_{\text{tct}_I} \geq \text{tct}_Y\) from (2) and replace \(\min_{\text{tct}_{I,O}} \geq \text{tct}_X\) with \(\min_{\text{tct}_I} \geq \text{tct}_X\) in (3). However, a composition of such components, while satisfying (2) and (3), may not satisfy the stronger versions of these conditions.

Consider single-channel \([I, O]\)-systems satisfying the persistence and liveness conditions; an elastic consumer is a \([\{Z\}, \emptyset]\)-system \(C\) satisfying (4) below; similarly, an elastic producer is a \([\emptyset, \{Z\}]\)-system \(P\) satisfying (5) and (6).

\[
C \models G \, \neg \text{stop}_Z \quad (4)
\]

\[
P \models G (\text{valid}_Z \land \text{stop}_Z \Rightarrow (\text{valid}_Z)^+) \quad (5)
\]

\[
P \models G \, \neg \text{valid}_Z \quad (6)
\]

Let \(C_Z\) be the \([Z, \text{valid}_Z, \text{stop}_Z]\)-system characterized by condition (4)—the largest (in the sense of behavior inclusion) of the systems satisfying this condition. Similarly, let \(P_Z\) be the \([Z, \text{valid}_Z, \text{stop}_Z]\)-system characterized by properties (5) and (6). Finally, define the \([I, O]\)-elastic environment to be the system

\[
\text{Env}_{I,O} = \bigcup_{X \in I} P_X \cup \bigcup_{Y \in O} C_Y.
\]

Note that \(\text{Env}_{I,O}\) is only a system; it is not functional and so is not a machine.

When a system satisfying the persistence and liveness conditions (1-3) is coupled with a matching elastic environment, the transfer on all data wires never comes to a stall:

**Lemma 3 (Liveness):** If \(S\) satisfies (1-3), then for every behavior \(\omega\) of \(S \cup \text{Env}_{I,O}\), all the component sequences of the transfer behavior \(\omega^T\) are infinite.

As an immediate consequence of Liveness Lemma, if \(S\) satisfies (1-3), then

\[
S^T = \{\omega^T \mid \omega \in S \cup \text{Env}_{I,O}\}
\]

is a well-defined \((I, O)\)-system.

**Definition 8:** An \([I, O]\)-system \(S\) is an \([I, O]\)-elastic machine if it satisfies the properties (1-3) and the associated system \(S^T\) is deterministic.

The liveness conditions (2,3) are visibly related to causality at the transfer level: \(k\) transfers on the output channels imply \(k\) transfers on the output channels in the cooperating environment. Thus, it is not surprising (even though the proof is not obvious) that the determinism postulated in Definition 8 suffices to derive the causality of \(S^T\):

**Theorem 2:** If \(S\) is an \([I, O]\)-elastic machine, then \(S^T\) is an \((I, O)\)-machine.

In the situation of Definition 8, we say that \(S\) is an elasticization of \(S^T\) and that \(S^T\) is the transfer machine of \(S\).

## IV. Elastic Networks

An elastic network \(N\) is given by a set of elastic machines \(S_1, \ldots, S_m\) with no shared wires, together with a set of channel pairs \((X_1, Y_1), \ldots, (X_n, Y_n)\), where the \(X_i\) are \(n\) distinct input channels and the \(Y_i\) are \(n\) distinct output channels. As a network of standard machines, the elastic network \(N\) is defined by

\[
N = \langle S_1, \ldots, S_m \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle.
\]

We will define a graph that encodes the sequentiality information about the network \(N\) and prove in Theorem 4 that acyclicity of that graph implies that \(N\) is an elastic machine and that \(N^\tau = \langle S_1^\tau, \ldots, S_m^\tau \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle\).

### A. Elastic Feedback

**Elastic feedback** is a simple case of elastic network:

\[
\langle S \mid P = Q \rangle = \langle S \mid P = Q, \text{valid}_P = \text{valid}_Q, \text{stop}_P = \text{stop}_Q \rangle.
\]

**Definition 9:** Suppose \(S\) is an elastic machine. An input-output channel pair \((P,Q)\) will be called sequential for \(S\) if

\[
S \models G \left( \min_{\text{tct}_{I\cup O}} = \text{tct}_Q \land \min_{\text{tct}_{I-(P)}} > \text{tct}_Q \Rightarrow \text{F valid}_Q \right). \quad (7)
\]

Condition (7) is a strengthening of the liveness condition (2) for channel \(Q\). It expresses a degree of independence of the output channel \(Q\) from the input channel \(P\); e.g., the first token at \(Q\) need not wait for the arrival of the first token at \(P\). This independence can be achieved in the system by storing some tokens inside, between these two channels. Note that (7) does not guarantee that connecting channels \(P\) and \(Q\) would not introduce ordinary combinational cycles. Therefore the acyclicity condition in the following theorem is required to ensure (by Theorem 1) that the elastic feedback, viewed as an ordinary network, is a machine.

**Theorem 3:** Let \(S\) be an elastic machine and \(F\) the elastic feedback system \(\langle S \mid P = Q \rangle\). If the channel pair \((P, Q)\) is sequential for \(S\), then: (a) the wire pair \((P, Q)\) is sequential for \(S^T\). If, in addition, \(\Delta(F)\) is acyclic, then: (b) \(F\) is an elastic machine, and (c) \(F^\tau = \langle S^\tau \mid P = Q \rangle\).
B. Main Theorems

Sequentiality of two channel pairs \((P, Q), (P', Q')\) of an elastic machine does not imply their “simultaneous sequentiality”

\[
S |\leftarrow G \left( \begin{array}{c}
\min\text{tct}_{I \cup O} = \text{cct}_{Q} \\
\min\text{tct}_{I-} > \text{cct}_{Q} \Rightarrow \text{F.valid}_{Q}
\end{array} \right).
\]

This deviates from the situation with ordinary machines, where the analogous property holds and is instrumental in the proof of Combinational Loop Theorem.

To justify multiple feedback on elastic machines, we have thus to postulate that simultaneous sequentiality is true where required. Specifically, we demand that elastic machines come with simultaneous sequentiality information: If \(S\) is an \([I, O]-\) elastic machine, then for every \(Y \in O\) a set \(\delta(Y) \subseteq I\) is given so that

\[
S |\leftarrow G \left( \begin{array}{c}
\min\text{tct}_{I \cup O} = \text{cct}_{Y} \\
\min\text{tct}_{I-} > \text{cct}_{Y} \Rightarrow \text{F.valid}_{Y}
\end{array} \right).
\] (8)

Note that if \(P \in \delta(Q)\), then the pair \((P, Q)\) is sequential, but the converse is not implied. A function \(\delta: O \rightarrow 2^I\) with the property (8) will be called a sequentiality interface for \(S\).

For an \([I, O]-\) elastic machine \(S\) with a sequentiality interface \(\delta\), we define \(\Delta^v(S, \delta)\) to be the graph with the vertex set \(I \cup O\) and directed edges \((X, Y)\) where \(X \notin \delta(Y)\). By Theorem 3(a), \(\Delta^v(S, \delta)\) contains \(\Delta(S)\) as a subgraph.

Given an elastic network \(N = \langle S_1, \ldots, S_n | X_1 = Y_1, \ldots, X_n = Y_n \rangle\), where each \(S_i\) comes equipped with a sequentiality interface \(\delta_i\), its graph \(\Delta^v(N)\) is by definition the direct sum of graphs \(\Delta^v(S_1, \delta_1), \ldots, \Delta^v(S_n, \delta_n)\) with each vertex \(X_i (1 \leq i \leq n)\) identified with the corresponding vertex \(Y_i\).

**Theorem 4:** If the graphs \(\Delta(N)\) and \(\Delta^v(N)\) are acyclic, then the network system \(N\) is an elastic machine, the corresponding non-elastic system \(\overline{N}\) is defined as \(\overline{S_1, \ldots, S_n | X_1 = Y_1, \ldots, X_n = Y_n}\) is a machine, and \(\overline{N}^* = \overline{N}\).

As in Theorem 3, acyclicity of \(\Delta(N)\) is needed to ensure (by Theorem 1) that \(\overline{N}\) defines a machine. Elasticization procedures (e.g. \([4]\)) will typically produce elastic components with enough sequential input-output wire pairs, so that \(\Delta(N)\) will be acyclic as soon as \(\Delta^v(N)\) is acyclic.

Note, however, that cycles in \(\Delta^v(N)\) need not correspond to combinational cycles in \(N\) seen as an ordinary network, since empty buffers with sequential elements cutting the combinational feedbacks may be inserted into \(N\). Even though non-combinational in the ordinary sense, these cycles contain no tokens and therefore no progress along them can be made.

Theorem 4 implies that insertion of empty elastic buffers does not affect the basic functionality of an elastic network, as illustrated in Figure 2(b).

**Definition 10:** An empty elastic buffer is an elastic machine \(S\) such that \(S^* = \text{Conn}(X, Y)\) for some \(X, Y\).

**Theorem 5 (Buffer Insertion Theorem):** Suppose \(B\) is an empty elastic buffer with channels \(X, Y\). Let \(N = \langle S_1, \ldots, S_n | X_1 = Y_1, \ldots, X_n = Y_n \rangle\) and \(M = \langle B, S_1, \ldots, S_n | X = Y_1, X_1 = Y, X_2 = Y_2, \ldots, X_n = Y_n \rangle\). If \(\Delta(N), \Delta(M), \text{ and } \Delta^v(N)\) are acyclic, then \(M\) is an elastic machine, and \(M^* = N^*\).

The precise relationship between graphs \(\Delta(M)\) and \(\Delta(N)\) can be easily described. In practice they are at the same time acyclic or not, as a consequence of sequentiality of sufficiently many input-output wire pairs of \(B\).

V. ELASTIC VS. PATIENT SYSTEMS

Elastic machines and patient processes of \([2]\) provide two formalizations of the intuitive concept of latency-insensitive circuits. In this section we address their connections and differences. We begin with an overview of \([2]\), using a minimalistic approach and terminology that differs from the original. We believe, however, that Definition 11 below matches the original definition accurately in most important aspects.

A. Patent Systems

The notation \(A^*\) is for the set of finite sequences over \(A\). A finitary \(W\)-system, by definition, is a set of behaviors \(\sigma\) such that \(\sigma.w\) is a finite sequence for every \(w \in W\).

A stalling stream over \(A\) is a stream over \(A \cup \{\varnothing\}\). We will refer to \(\varnothing\) as the bubble and to elements of \(A\) as tokens. We will consider only stalling streams that contain finitely many tokens. If \(a\) is such a stream, let \(\pi \in A^*\) denote the sequence over \(A\) obtained by dropping all bubbles from \(a\). Clearly, \(a\) is determined by \(\pi\) and the sequence \(\partial(a) = N^*\) of lengths of bubble sequences between consecutive tokens of \(a\). For example, if

\[
a = (\varnothing, \varnothing, 7, 4, 5, \varnothing, \varnothing, \varnothing, 8, \ldots)\]

we have \(\pi = (7, 4, 5, 8, \ldots)\) and \(\partial(a) = (2, 1, 0, 3, \ldots)\). Two stalling streams \(a, b\) are latency equivalent, written \(a \preceq b\), when \(\pi = \bar{b}\). Note that \(a \preceq \pi\).

By definition, a stalling \(W\)-system is a set of behaviors \(\sigma\) such that for every \(w \in W\), \(\sigma.w\) is a stalling stream over \(\text{type}(w)\). Latency equivalence extends to \(W\)-behaviors and \(W\)-systems: \(\sigma \preceq \tau\) iff \(\sigma.w \preceq \tau.w\) holds for every \(w \in W\); \(\sigma \preceq \tau\) iff for every \(\sigma \in E\) \((\sigma \in S')\) there exists \(\tau \in S\) \((\tau \in S')\) such that \(\sigma \preceq \tau\).

A stalling \(W\)-system \(S\) determines a standard finitary \(W\)-system \(S^\tau = \{\sigma | \sigma \in S\}\), where \(\sigma\) is given by \(\sigma.w = \sigma.\bar{w}\) (for all \(w \in W\)). Clearly, \(S^\tau \preceq S\).

Stalling the \(k\)-th token of \(a\) by \(d\) steps produces a latency equivalent stream that will be denoted \(\text{stall}(a, k, d)\). Omitting the easy definition, we give an example: if \(a\) is as in (9), then

\[
\text{stall}(a, 1, 3) = (\varnothing, \varnothing, 7, \varnothing, \varnothing, \varnothing, 4, 5, \varnothing, \varnothing, \varnothing, 8, \ldots)
\]

**Definition 11:** Let \(\prec\) be a well-founded order\(^1\) on \(W\) and let \(D > 0\). A patient \(W\)-system (relative to \(\prec\) and \(D\)) is a

\(^1\)Introduction of a well-founded ordering of wires is motivated in \([2]\) with the purpose of modeling combinational dependencies, but such dependencies in patient systems are not discussed in any detail. Moreover, the ordering of wires is implicitly assumed to be total in \([2]\), which is somewhat unnatural. For instance, when constructing a patient adder with inputs \(u, v\) and output \(w\), one has two ordering choices: \(u \prec_1 v \prec_1 w\) and \(v \prec_2 u \prec_2 w\). It is not clear that a patient adder in the \(\prec_1\)-sense will be patient in the \(\prec_2\)-sense too.
stalling system \( \mathcal{P} \) such that for every \( \sigma \in \mathcal{P} \), every \( u \in W \), and every \( k \geq 0 \) there exists \( \sigma' \in \mathcal{P} \) such that
\[
\text{(Pat-1)} \quad \sigma'.u = \text{stall}(\sigma, u, k, 1)
\]
and for every \( v \neq u \) there exists \( d_v \leq D \) such that
\[
\text{(Pat-2)} \quad \sigma'.v = \begin{cases} \text{stall}(\sigma, v, k, d_v) & \text{if } u \prec v \\ \text{stall}(\sigma, v, k + 1, d_v) & \text{otherwise} \end{cases}
\]
The main results of [2] can now be summarized:
1) a theorem saying that the composition of patient systems (with the same \( W \), \( \prec \), and \( D \)) is a patient system;
2) the definition and analysis of patient buffers, i.e. patient systems \( \mathcal{B} \) such that \( \mathcal{B}^\text{fin} = \text{Conn}^\text{fin}(u, v) \) — the finitary connection system;
3) a general construction that, for a given finitary system \( \mathcal{M} \) without combinational dependencies (model of a Moore machine), produces a patient system \( \mathcal{P} \) such that \( \mathcal{P} \cong \mathcal{M} \).

### B. Comparison

The formalization given by patient systems is at a higher level of abstraction. While elastic machines deal explicitly with handshaking signals between communicating systems, patient systems communicate purely in the token/bubble language.

Given an elastic (as defined in Section III) \((I, O)\)-system \( \mathcal{E} \), the corresponding stalling \((I \cup O)\)-system \( \mathcal{E}^\Delta \) is obtained by projecting the finite-transfer behaviors of \( \mathcal{E} \) to data wires and replacing data items on each wire with \( \Box \) at all cycles where transfer along that wire does not occur. Precisely, let \( \mathcal{E}^\Delta \) be the subset of \( \mathcal{E} \) consisting of all behaviors \( \omega \) such that \( \omega^\Delta \) is finite for all channels \( Z \).\(^2\) Then, given \( \omega \in \mathcal{E}^\Delta \), we define a stalling \((I \cup O)\)-behavior \( \omega^\Delta \) by
\[
(\omega^\Delta, Z)[i] = \begin{cases} (\omega.Z)[i] & \text{if } (\omega.\text{valid}_{Z})[i] \land \neg(\omega.\text{stop}_{Z})[i] \\ \Box & \text{otherwise} \end{cases}
\]
and finally we define the stalling system \( \mathcal{E}^\Delta \) as the set of all such behaviors \( \omega^\Delta \). Clearly, the system \( \mathcal{E}^\Delta \) is the finitary version of the standard machine \( \mathcal{E}^\Delta \).

Now we can address some questions pertinent to the comparison of patient processes vs. elastic machines.

**Are patient processes more general?** The answer is “no” because there exist elastic machines \( \mathcal{E} \) such that \( \mathcal{E}^\Delta \) is not patient. To see this, consider an elastic machine \( \mathcal{E} \) that starts offering new valid outputs on channel \( u \) only on even cycles. (The existence of such elastic machines is obvious.) Observe that \( \sigma.u = (\Box, 7, 9, \ldots) \) is possible for some behavior \( \sigma \) of \( \mathcal{E}^\Delta \) (token 7, even though transmitted on cycle 1 was first offered on cycle 0). Then \( \text{stall}(\sigma, u, 0, 1) = (\Box, \Box, 7, 9, \ldots) \) must also be part of a behavior of \( \mathcal{E}^\Delta \), by condition (Pat-1) of Definition 11. This implies that token 9 is first offered on cycle 3, contrary to our assumption.

The above example can be viewed as an indication that the condition (Pat-1) is too restrictive. It would be interesting to see if an appropriate modification of (Pat-1) results in a definition of patient processes that captures elastic machines.

\(^2\)One can prove that \( \mathcal{E} \) is the set of all limits of behaviors of \( \mathcal{E}^\Delta \) and so \( \mathcal{E} \) is determined by \( \mathcal{E}^\Delta \).

**Are elastic machines more general?** The answer is an easy “no” since, for example, the set of all possible stalling \( W \)-behaviors is a patient system in the sense of Definition 11. However, if one adds to Definition 11 a reasonable requirement that a patient system be a machine, the answer is not immediately clear.

**Which formalization is easier to use?** Without offering a definitive answer, we would argue that verifying that a low-level design (RTL, say) implements an elastic machine would be easier than verifying that it implements a patient system. The bottom line is that the conditions for a system to be an elastic machine are expressible as temporal properties of suitably constructed infinite-state models. This is not obvious for the determinism condition for \( \mathcal{S}^\Delta \) in Definition 8, but can be done by replacing determinism with causality and introducing auxiliary variables for sequences of transferred values over channels. Even though (e.g., because of infinite counters involved) these conditions are not directly checkable by the existing model checking technology, there are palatable opportunities to find manageable stronger conditions that taken together imply elasticity (e.g., postulating a limit on the token count differences between channels eliminates the need for infinite counters). On the other hand, the definition of a patient system, being of the form “for every behavior \( \sigma \), there exists a behavior \( \sigma' \) such that . . .” appears to us to be intrinsically more complex. Our only positive conclusion, however, is that the mechanical checking of either of the definitions is an open problem deserving further study.

### VI. Conclusion

We have presented a theory of elastic machines that gives an easy-to-check condition for the compositional theorem of the form “an elasticization of a network of ordinary components is equivalent to the network of components’ elasticizations”. Verification of a particular implementation is reduced to proving that conditions of Definition 8 are satisfied for all elastic components used, and that the graph \( \Delta^e(N^e) \) is acyclic for every network \( N \) to which the elasticization is applied. While the definition of the graphs \( \Delta^e \) may appear complex because of the sequentiality interfaces involved, it should be noted that the elasticization procedures, e.g. [4], are reasonably expected to completely preserve sequentiality: a channel \( P \) belongs to \( \delta(Q) \) if the wire-pair \((P, Q)\) is sequential in the original non-elastic machine. This ensures \( \Delta^e(N^e) = \Delta(N) \) and so testing for sequentiality is done at the level of ordinary networks.

Future work will be focused on proving correctness of particular elasticization methods, on techniques for mechanical verification of elasticity, and on extending the theory to more advanced protocols.

**Acknowledgments:** Luca Carloni clarified some details of [2]. Ken McMillan pointed out several inaccuracies in a previous version of the paper and further clarified [2] for us. Gerard Berry, Ching-Tsun Chou, John Harrison, and the anonymous reviewers provided useful remarks. We are grateful for all the help we received.
REFERENCES


Proof of Lemma 1

Remark. Lemma 1 and Lemma 4 below are well-known facts. We give their proofs for the sake of completeness.

Suppose a and b are both fixpoints of f. By the contraction property, \( a \sim_k b \Rightarrow a \sim_{k+1} b \) holds for every k. Since a \( \sim_0 b \), it follows that a \( \sim_k b \) holds for every k, so a = b. This proves the uniqueness part.

Suppose \( p \) is any finite sequence of length \( k \) and let \( S_p \) be the set of all streams \( a \in A^\infty \) such that \( p \) is a prefix of a. It follows from the contraction condition that all streams in the set \( \{ f(a) \mid a \in S_p \} \) have a common prefix of length \( k+1 \). We will denote that common prefix by \( p^! \). Thus, for every \( p \in A^* \), there exists \( p^! \in A^* \) such that \( |p^!| = 1 + |p| \) and
\[
\forall a \in A^\infty. \, p \preceq a \Rightarrow p^! \preceq f(a),
\]
where \( \preceq \) stands for the prefix relation. This implies
\[
\forall p, q \in A^*. \, p \preceq q \Rightarrow p^! \preceq q^!.
\]
Consider now the sequence \( \epsilon, \epsilon^!, \epsilon^{\ast}, \ldots \), where \( \epsilon \) is the empty sequence. Since \( \epsilon \preceq \epsilon^! \preceq \epsilon^{\ast} \preceq \cdots \preceq \epsilon \), we obtain \( \epsilon \preceq \epsilon^! \preceq \epsilon^{\ast} \preceq \cdots \). Since the sequences in this chain have increasing length, there is a unique “limit” \( a \in A^\infty \) satisfying \( \epsilon \preceq \epsilon^! \preceq \epsilon^{\ast} \preceq \cdots \preceq a \). By (10), the relations \( \epsilon \preceq \epsilon^! \preceq \epsilon^{\ast} \preceq \cdots \preceq \epsilon \), imply \( \epsilon \preceq f(a), \epsilon^! \preceq f(a), \epsilon^{\ast} \preceq f(a), \ldots \) for all \( a \in A^\infty \). Thus, \( \epsilon \preceq \epsilon^! \preceq \epsilon^{\ast} \preceq \cdots \preceq f(a) \), which forces \( a = f(a) \) (by the uniqueness of the “limit” stream), proving the existence of the fixpoint. This finishes the proof of Lemma 1.

The following fact will be used in the proof of Lemma 2. To state it, we need a definition: A sequence \( a_n \) of elements of \( A^\infty \) (i.e. a sequence of streams over \( A \)) converges to \( a \in A^\infty \) if every prefix of \( a \) is the prefix of all but finitely many of the \( a_n \).

\[
\text{Lemma 4: If } f : A^\infty \to A^\infty \text{ is contractive and } \bar{a} \text{ is an arbitrary element of } A^\infty \text{ then the sequence of iterations } f^n(\bar{a}) \text{ (} n \geq 0 \text{) converges to the fixpoint of } f.
\]

Proof. By Lemma 1, f has a unique fixpoint \( a \). We prove that \( f^n(\bar{a}) \sim_n a \) holds for every \( m, n \) such that \( m \geq n \). (This clearly implies the lemma.) Our statement is obvious for \( n = 0 \). Arguing by induction on \( n \), for \( n + 1 \) we have \( f^{n+1}(\bar{a}) \sim_{n+1} a \) (induction hypothesis), so applying f to both sides and using the contraction property and \( f(a) = a \) we derive \( f^n(\bar{a}) \sim_n a \).

B. Proof of Lemma 2

Sequentiality of the pair \((u, v)\) implies that for every \( \sigma \in \{ I - \{ u \} \} \), there exists a unique fixpoint \( a_\sigma \) of \( F^u_{uv} \). (We continue with the notation from the paragraph preceding the lemma.) Consequently, \( F(\sigma * [u \mapsto a]) \) has the form \( \tau * [v \mapsto a] \) for a unique \( \tau \in \{ O - \{ v \} \} \). Let \( G : \{ I - \{ u \} \} \to \{ O - \{ v \} \} \) be the function that associates \( \tau \) with \( \sigma \) in the way just described. It is easy to verify (by definition of \( G \)) that
\[
\sigma * \tau \in \langle S \mid u = v \rangle \iff G(\sigma) = \tau
\]
holds for every \( \sigma \in \{ I - \{ u \} \}, \tau \in \{ O - \{ v \} \} \), so it only remains to prove that \( G \) satisfies the causality condition. That in turn is a simple consequence of the causality of \( F \) and the causality of the parametrized fixpoint operator that associates \( a_\sigma \) with \( \sigma \). By the latter we mean that \( a_\sigma \sim_k a_{\sigma'} \) always follows from \( \sigma \sim_k \sigma' \). This we state now and prove as a separate lemma.

Lemma 5: If \((u, v)\) is a sequential pair for \( F, \sigma \sim_k x' \), and \( a_\sigma \) and \( a_{\sigma'} \) are the fixpoints of \( F^x_{uv} \) and \( F^x_{v'v} \), then \( a_\sigma \sim_k a_{\sigma'} \).

Proof. Let \( A = \llbracket u \rrbracket = [v] \) and let \( H : \llbracket I - \{ u \} \rrbracket \times A \to A \) be given by \( H(\sigma, a) = F(\sigma * [u \mapsto a]) \). Our sequentiality assumption reads as follows:
\[
(12) \forall \sigma, \sigma', a, a'. \, \sigma \sim_{k+1} \sigma' \land a \sim_k a' \Rightarrow H(\sigma, a) \sim_k H(\sigma', a')
\]
Starting with an arbitrary \( \bar{a} \in A \), define
\[
a_n = \begin{cases} a & \text{if } n = 0 \\ H(\sigma, a_{n-1}) & \text{if } n \geq 1 \end{cases}
\]
\[
a'_n = \begin{cases} a & \text{if } n = 0 \\ H(\sigma', a'_{n-1}) & \text{if } n \geq 1 \end{cases}
\]
Since \( a_\sigma \) is the fixpoint of the function \( F^x_{uv} : \bar{a} \mapsto H(\sigma, a) \), we have (by Lemma 4) that \( a_\sigma = \lim a_n \), and similarly \( a_{\sigma'} = \lim a'_n \). By induction, it follows from (12) that \( a_\sigma \sim_k a_n \) and \( a_{\sigma'} \sim_k a'_n \) hold for every \( n \). Also by induction, it follows from (12) that \( a_n \sim_k a'_n \) holds for all \( n \leq k \). Therefore, \( a_\sigma \sim_k a_{\sigma'} \), as required.

C. Běkić Lemma

[Bibliographical information. Běkić Lemma asserts that the fixpoint of a two-variable function can be computed by iterating the fixpoint operators along the two coordinates. It holds in various contexts. The original result is in: Hans Běkić, Definable operations in general algebras, and the theory of...]

10

We begin with a generalization of the concept of sequentiality.

**Definition 12:** Let $S$ be an $(I, O)$-machine given by $F$: $[I] \rightarrow [O]$ and let $A \subseteq I$, $v \in O$. The pair $(A, v)$ is sequential for $S$ if for every $k \geq 0$, every $\sigma, \sigma' \in [I - A]$, and every $\tau, \tau' \in [A]$ one has

$$\sigma \sim_{k+1} \sigma \wedge \tau \sim_k \tau' \Rightarrow F(\sigma \ast \tau).v \sim_{k+1} F(\sigma' \ast \tau').v.$$  

**Lemma 6:** $(A, v)$ is sequential if and only if $(u, v)$ is sequential for every $u \in A$.

**Proof.** The “only if” part is trivial. We prove the “if” part for the case when $A$ has two elements. The general case is only notationally more difficult.

Thus, suppose $A = \{x, y\}$ and both $(x, v)$ and $(y, v)$ are sequential. We need to prove

$$\sigma \sim_{k+1} \sigma \wedge a \sim_k a' \wedge b \sim_k b' \Rightarrow F(\sigma \ast [x \mapsto a] \ast [y \mapsto b]).v$$  

$$\sim_{k+1} F(\sigma' \ast [x \mapsto a'] \ast [y \mapsto b']).v.$$  

For the proof, just observe that both $F(\sigma \ast [x \mapsto a] \ast [y \mapsto b]).v$ and $F(\sigma' \ast [x \mapsto a'] \ast [y \mapsto b']).v$ are equivalent in the sense of $\sim_{k+1}$ with $F(\sigma' \ast [x \mapsto a'] \ast [y \mapsto b]).v$, as a consequence of the assumed sequentiality of $(x, v)$ and $(y, v)$.

**Lemma 7 (Bekić Lemma):** Suppose $(u, w)$ and $(v, z)$ are sequential pairs for a machine $S$ and suppose that the four wires involved are distinct. If one of the pairs $(u, z)$, $(v, w)$ is also sequential, then $(v, z)$ is sequential for $\langle S | u = w \rangle$, $(u, w)$ is sequential for $\langle S | v = z \rangle$, and (therefore) the system $\langle S | u = w, v = z \rangle$ is a machine.

**Proof.** In view of Lemma 6, it suffices to prove that $(v, z)$ is sequential for $\langle S | u = w \rangle$ under either of the following two assumptions:

(i) $(u, w)$ and $(\{u, v\}, z)$ are sequential;

(ii) $(v, z)$ and $(\{u, v\}, w)$ are sequential.

Thus, our goal is to prove

$$\sigma \sim_{k+1} \sigma' \wedge b \sim_k b'$$  

$$\Rightarrow G(\sigma \ast [v \mapsto b]).z \sim_{k+1} G(\sigma' \ast [v \mapsto b']).z$$  

where $G$ is the function corresponding to the system $\langle S | u = w \rangle$. We can restate this in a more convenient form

$$\sigma \sim_{k+1} \sigma' \wedge b \sim_k b'$$  

$$\wedge G(\sigma \ast [v \mapsto b]) = \tau \ast [z \mapsto c]$$  

$$\wedge G(\sigma' \ast [v \mapsto b']) = \tau \ast [z \mapsto c']$$  

$$\Rightarrow c \sim_{k+1} c'$$  

By definition of $G$, this can be further restated as

$$\sim_{k+1} \sigma' \wedge b \sim_k b' \wedge a \sim_k a'$$  

$$\wedge F(\sigma \ast [v \mapsto b] \ast [u \mapsto a]) = \tau \ast [z \mapsto c] \ast [w \mapsto a]$$  

$$\wedge F(\sigma' \ast [v \mapsto b'] \ast [u \mapsto a']) = \tau \ast [z \mapsto c'] \ast [w \mapsto a']$$  

$$\Rightarrow c \sim_{k+1} c'.$$  

We finish the proof now by deriving (13) from any of our two assumptions (i), (ii). If (i) holds, then (13) immediately follows. If (ii) holds, we first derive $a \sim_{k+1} a'$ from the assumptions of (13) and sequentiality of $(\{u, v\}, w)$. Then with $a \sim_{k+1} a'$ in place of $a \sim_k a'$ in the assumptions of (13), the conclusion $c \sim_{k+1} c'$ follows by sequentiality of $(v, z)$.

**D. Proof of Theorem 1**

The notation $\Gamma_{[u_1 = v_1, \ldots, u_n = v_n]}$ will stand for the quotient graph obtained by identifying the vertices $u_i$ and $v_i$ ($i = 1, \ldots, n$) in $\Gamma$.

For a given directed graph $\Gamma$, a source vertex $u$ of $\Gamma$, and a sink vertex $v$ of $\Gamma$, let $\Gamma_{uv}$ denote the graph obtained from $\Gamma$ by:

1. removing $u, v$ and all edges incident with them;
2. adding an edge $(w, z)$ for all $w, z$ such that $(w, u)$ and $(v, w)$ are edges of $\Gamma$.

(See Figure 6.)

**Definition 13:** If $\Gamma$ and $\Gamma'$ are directed graphs such that every vertex of $\Gamma$ is a vertex of $\Gamma'$ and every two vertices joined by an edge in $\Gamma$ are joined by a path in $\Gamma'$, then we say that $\Gamma$ is immersed in $\Gamma'$ and write $\Gamma \rightarrow \Gamma'$.

In the main text, we used the same notation $\Delta$ for dependency graphs of machines and systems. This is slightly ambiguous, however. For example, if $S$ is an $(I, O)$-machine, then $N = \langle S | u = v \rangle$ is a system, but at the same time an $(I - \{u\}, O - \{v\})$-machine. When $N$ is regarded as a network, then $\Delta(N)$ is the quotient graph of $\Delta(S)[u = v]$. When $N$ is regarded as a machine, the vertex set of $\Delta(N)$ is $I \cup O - \{u, v\}$. For the proof, we need to disambiguate this notation.

**Definition 14:** The dependency graph $D(S)$ of an $(I, O)$-machine $S$ has $I \cup O$ as the vertex set, and $\{(u, v) \in I \times O | \langle u, v \rangle$ is not sequential for $S\}$ as its set of directed edges. The dependency graph $\Delta(N)$ of a network $N = \langle S_1, \ldots, S_m \rangle$ is $(D(S_1) \cup \cdots \cup D(S_m))[u_1 = v_1, \ldots, u_n = v_n]$.  

**Definition 15:** We write $N \sim N'$ when the networks $N$ and $N'$ describe the same system.

To appreciate Definition 15, note that networks are a form of system descriptions, so that, strictly speaking, a network

A source vertex is a vertex with no incoming edges; a sink vertex is a vertex with no outgoing edges.
is not a system, although it uniquely determines one. The equivalence $\mathcal{N} \sim \mathcal{N}'$ implies $D(\mathcal{N}) = D(\mathcal{N}')$ but it does not imply $\Delta(\mathcal{N}) = \Delta(\mathcal{N}')$.

**Lemma 8:** If $u, v$ are vertices of $\Gamma$ and if $\Gamma \rightarrow \Gamma'$, then $\Gamma[u=v] \rightarrow \Gamma'[u=v]$.

**Proof.** Trivial. □

**Lemma 9:** If $u$ is a source of $\Gamma$ and $v$ is a sink of $\Gamma$, then $\Gamma[u-v] \rightarrow \Gamma'[u=v]$.

**Proof.** Easy. (Use Figure 6.) □

**Lemma 10:** If $(u, v)$ is sequential for $S$, then $D(\langle S | u = v \rangle) \rightarrow D(S)$, and $D(S[\{u=v\}]) \rightarrow D(S)$ in $\mathcal{N}$. The second of these relations is an instance of Lemma 9. For the first, we prove that $D(\langle S | u = v \rangle)$ is a subgraph of $D(S)[u=v]$. Notice that these graphs have a common set of vertices $I \cup O \setminus \{u, v\}$. Suppose $(w, z)$ is an edge of $D(\langle S | u = v \rangle)$, i.e., $(w, z)$ is not sequential in $\langle S | u = v \rangle$. By Bekić Lemma, this implies that either $(w, z)$ is not sequential in $S$ or that both $(w, v), (u, z)$ are not sequential in $S$. In both cases, it follows that $(w, z)$ is an edge in $D(S)$.

We strengthen now Theorem 1 as follows:

If $\mathcal{N} = \langle S_1, \ldots, S_m | u_1 = v_1, \ldots, u_n = v_n \rangle$ and $\Delta(\mathcal{N})$ is acyclic, then $\mathcal{N}$ is a machine and $D(\mathcal{N}) \rightarrow \Delta(\mathcal{N})$.

We prove this strengthened theorem by induction on $n$. The case $n = 0$ amounts to the fact mentioned in the main text: if $S_1, \ldots, S_m$ are machines with disjoint wire sets, then $S = S_1 \cup \cdots \cup S_m$ is a machine and an input-output pair $(u, v)$ is sequential for $S$ if either (1) $(u, v)$ is sequential for some $S_i$, or (2) $u$ and $v$ belong to different machines $S_i, S_j$.

For the induction step, let $\mathcal{N}' = \langle S_1, \ldots, S_m | u_1 = v_1, \ldots, u_{n-1} = v_{n-1} \rangle$. By induction hypothesis, $\Delta(\mathcal{N})$ is a machine, so we can write $\mathcal{N} \sim \langle S'_1, \ldots, S'_m | u_n = v_n \rangle$. Note that $\Delta(\mathcal{N}) = \Delta(\mathcal{N'})[u_n = v_n]$. It follows from this and the induction hypothesis $D(\mathcal{N}') \rightarrow \Delta(\mathcal{N}')$ that $D(\mathcal{N})[u_n = v_n] \rightarrow \Delta(\mathcal{N})$. This in turn, together with acyclicity of $\Delta(\mathcal{N})$, implies that $(u_n, v_n)$ is a sequential pair for $\mathcal{N}'$. Thus, by Lemma 2, $\mathcal{N}$ is a machine. Finally,

$$D(\mathcal{N}) = D(\mathcal{N'})[u_n = v_n] \rightarrow \Delta(\mathcal{N})$$

finishing the proof. The first immersion is justified by Lemma 10 and the second by induction hypothesis together with Lemma 8.

### E. Handshake Lemma

Persistence conditions (1) will be used through the following easily derived consequence.

**Lemma 11 (Handshake):** If $S$ satisfies (1), then, for every $Y \in O$,

$$S \models GF \top \land GF \not\top Y \Rightarrow GF \text{transfer}_Y$$

**Proof.** From (1) we obtain

$$S \models G(\text{valid}_Y \land F \not\top Y) \Rightarrow F \text{transfer}_Y$$

which implies the lemma. □

### F. Proof of Lemma 3

Assuming the contrary, let $\omega$ be a behavior of $S \sqcup Env_{I,O}$ and let $Z$ be a wire such that the transfer sequence $\omega \uparrow Z$ is finite and of shortest length. This assumption implies $\omega \models F \mathsf{G \min\text{tct}_{I,O}} \geq \mathsf{tct}_G$. If $Z \in I$, then from (3) we get $\omega \models GF \not\top Z$. We also have $\omega \models GF \text{valid}_Z$, by (6). Since elastic producers satisfy the persistence condition, Handshake Lemma implies $\omega \models GF \text{transfer}_Z$, which is a contradiction.

There remains the case $Z \in O$, where we can also assume that there are no $X \in I$ with $\omega \uparrow X$ of the same length as $\omega \uparrow Z$. From this additional assumption we have $\omega \models F GF \min\text{tct}_I > \mathsf{tct}_Z \land \mathsf{tct}_O \geq \mathsf{tct}_G$, so from (2) we get $\omega \models GF \text{valid}_Z$. A call to consumer liveness (4) and then to Handshake Lemma yields the contradictory conclusion $\omega \models GF \text{transfer}_Z$, as in the previous case.

### G. Preliminaries for Proof of Theorem 2

Define an *elastic k-producer* as a system $P$ that satisfies the producer persistence condition (5) and the following weakened form of the producer liveness condition (6):

$$P \models G(\text{tct}_Z < k \Rightarrow F \text{valid}_Z).$$

(14)

Thus, a $k$-producer promises to cooperate in creation of at least $k$ transfers.

Similarly, we weaken the definition of the systems $P_Z$ (for any wire $Z$) and define $P_Z^k$ as the system defined by conditions (5) and (14). Then we define the *k-environment* (a system, but not a machine)

$$\mathsf{Env}_{I,O}^k = \bigcup_{X \in I} P_X^k \sqcup \bigcup_{Y \in O} C_Y.$$

The following is a finite-transfer version of Lemma 3, proved in much the same way. Note that since $\mathsf{Env}_{I,O} \subseteq \mathsf{Env}_{I,O}^k$ for every $k$, Lemma 12 actually implies Lemma 3.

**Lemma 12:** Let $S$ be an $[I, O]$-system satisfying the conditions (1-3). Then for every behavior $\omega \in S \sqcup \mathsf{Env}_{I,O}^k$, all the component streams of the transfer behavior $\omega \uparrow Z$ have length at least $k$.

**Proof.** The proof of Lemma 3 applies almost verbatim. The only change is that, arguing by contraction, we can now assume that, in addition to being finite, the sequence $\omega \uparrow Z$ has length $< k$. Consequently, the intermediate result $\omega \models GF \text{valid}_Z$ is derived from (14), (6), and $\omega \models \mathsf{tct}_Z < k$. □

We will also need two easy lemmas about standard (non-elastic) machines.

**Lemma 13:** Recall that every network

$$\mathcal{N} = \langle S_1, \ldots, S_m | u_1 = v_1, \ldots, u_n = v_n \rangle$$

is the system obtained by hiding the wires $u_i$ and $v_i$ in the corresponding system

$$\mathcal{N}^k = S_1 \sqcup \cdots \sqcup S_m \sqcup \mathsf{Conn}(u_1, v_1) \sqcup \cdots \sqcup \mathsf{Conn}(u_n, v_n).$$
and that $W^2 = W \cup \{u_1, v_1, \ldots, u_n, v_n\}$ is the relationship between the wire sets of these systems. Suppose that the graph $\Delta(N)$ is acyclic. Then the map $\sigma \mapsto \sigma \downarrow W^2 : \mathcal{N^2} \to \mathcal{N}$ is a bijection. In particular, if $W = \emptyset$, then $\mathcal{N^2}$ has exactly one behavior.

**Proof.** Consider the case $n = 1$. For this case, our lemma says that when $S$, $u, v$ are as in Lemma 2, then for every behavior $\sigma \in (S | u = v)$ there is a unique $a \in \llbracket a \rrbracket$ such that $\sigma \ast [u \mapsto a \ast ] \ast [v \mapsto a] \in S$. This is clear from considerations in Section II-E, notably Lemma 2.

The general case follows by induction.

**Lemma 14:** Suppose $C$ is a machine with all input-output wire pairs sequential. Suppose also that in the network $N = \langle C, C' | X_1 = Y_1, \ldots, X_n = Y_n \rangle$ exactly one wire of each pair $(X_i, Y_i)$ is a wire of $C$ and the other is a wire of $C'$. Then $\Delta(N)$ is acyclic.

**Proof.** By assumption, the graph $D(C)$ has no edges. Thus, $\Delta(N)$ is the union of a bipartite graph $D(C')$ and a set of vertices, and so is acyclic.

**H. Proof of Theorem 2**

We will prove that $\mathcal{S}^T = \{\omega^T | \omega \in S \sqcup \Env_{1,0}\}$ is causal and accepts every input stream.

By assumption, the transfer determinism condition holds for $S$:

\[
\forall \zeta, \vartheta \in S \sqcup \Env_{1,0}. \quad \zeta^T.I = \vartheta^T.I \Rightarrow \zeta^T.O = \vartheta^T.O. \tag{15}
\]

Let us define a **partial $W$-behavior** (for any given set $W$ of wires) as a function $\theta$ that associates to every wire $w \in W$ a (finite or infinite) sequence $\theta.w \in \text{type}(w)^\ast$. Note that every behavior is a partial behavior as well. If all sequences $\theta.w$ ($w \in W$) have length $k$, we will say that $\theta$ is a partial $W$-behavior of length $k$. We allow $k = \infty$ in this definition, so that a partial behavior of length $\infty$ is just a behavior.

Our main goal—the causality of $\mathcal{S}^T$—will be proved when we have established the following equivalent property: For every partial $I$-behavior $\theta$ of finite length $k$, there exists a partial $O$-behavior $\chi$ of length $k$ such that

\[
\forall \vartheta \in S \sqcup \Env_{1,0}. \quad \theta \preceq \vartheta^T.I \Rightarrow \chi \preceq \vartheta^T.O. \tag{16}
\]

To prove this, we will need to put our system $S$ in “eager” environments that are themselves machines, defined next. The eager environments will let us define $\chi$ as a function of $\theta$; see (18) below. In the remainder of the proof, we will check that the so obtained $\chi$ satisfies (16).

For any wire $Z$ and a (finite or infinite) sequence $\alpha$ of elements of $\text{type}(Z)$, let $\mathcal{P}^\alpha_Z$ be the “eager producer” machine that offers transfer of elements of $\alpha$ in order, and if it succeeds in transferring them all, then it stops offering further transfer. As a system, it is characterized by the following properties:

\[
\mathcal{P}^\alpha_Z \models \text{G (tct}_Z = m \land m < |\alpha| \Rightarrow \text{valid}_Z \land Z = \alpha[m])
\]

\[
\mathcal{P}^\alpha_Z \models \text{G (tct}_Z \geq |\alpha| \Rightarrow \neg\text{valid}_Z \land Z = \text{Arb}_Z)
\]

where $|\alpha|$ denotes the length of $\alpha$ and $\text{Arb}_Z$ is an arbitrary element of $\text{type}(Z)$. Note that when $\alpha$ is infinite the second condition is vacuously true and the conjunct $m < |\alpha|$ can be removed from the first condition.

**Lemma 15:** For every behavior $\sigma \in \mathcal{P}^\alpha_Z$, the transfer sequence $\sigma^T.Z$ is a prefix of $\alpha$.

Define also the “eager consumer” machine $\mathcal{C}^\alpha_Z$ (for any wire $Z$), characterized by the property

\[
\mathcal{C}^\alpha_Z \models \text{G - stop}_Z.
\]

For any partial $I$-behavior $\theta$, define the system

\[
\mathcal{S}^\theta = S \sqcup \mathcal{P}^\theta \cup \bigcup_{Y \in \mathcal{O}} \mathcal{C}^\theta_Y, \quad \text{where} \quad \mathcal{P}^\theta = \bigcup_{X \in \mathcal{I}} \mathcal{P}^\theta_X. \tag{17}
\]

**Lemma 16:** Let $\theta$ be a partial $I$-behavior of length $k$. The system $\mathcal{S}^\theta$ has exactly one behavior $\omega$ and it satisfies $\omega^T.I = \theta$. Moreover, $|\omega^T.Y| \geq k$ for every $Y \in \mathcal{O}$.

**Proof.** It is easy to check that for finite sequences $\alpha$ one has $\mathcal{P}^\alpha_Z \subseteq \mathcal{P}_Z^{[\alpha]}$, while for infinite $\alpha$ one has $\mathcal{P}^\alpha_Z \subseteq \mathcal{P}_Z$. Consequently, if $\theta$ is a partial $I$-behavior of length $k$, we have that $\mathcal{S}^\theta$ is a subsystem of $S \sqcup \Env_{1,0}^k$, where we assume that $S \sqcup \Env_{1,0}^k$ is just $S \sqcup \Env_{1,0}$. By Lemma 12, $|\omega^T.Z| \geq k$ holds for all behaviors $\omega$ of $\mathcal{S}^\theta$ and all channels $Z$.

The eager producer and eager consumer machines have the property that their outputs depend sequentially on all their inputs. Note that $\mathcal{S}^\theta$ is of the form $N^\ast$ (see Lemma 13) for a network $N$ whose one component is $S$ and all other components are eager producers or consumers. Repeated application of Lemma 14 proves $\Delta(\mathcal{S}^\theta)$ is acyclic. Then Lemma 13 implies that $\mathcal{S}^\theta$ has a unique behavior $\omega$. Since $\omega$ is also a behavior of $\mathcal{P}^\theta$, we obtain from Lemma 15 that for every input channel $X$ the sequence $\omega^T.X$ is a prefix of $\theta.X$. Since $|\omega^T.X| \geq k$, this implies $\omega^T.X = \theta.X$ for every $X$, which is to say that $\omega^T.I = \theta$.

Lemma 16 implies immediately that for every $\theta \in [I]$ there exists $\omega \in S$ such that $\omega^T.I = \theta$. Thus, $\mathcal{S}^T$ accepts all inputs and it remains to prove that it satisfies the causality property (16). For this proof, we fix a partial $I$-behavior $\theta$ of finite length $k$ and let $\omega$ be as provided by Lemma 16. The lemma also implies the existence of a number $n$ such that $\omega[n] = \text{tct}_Z \geq k$ for all channels $Z$. Thus, there exist a partial $O$-behavior $\chi$ of length $k$ such that $\chi \preceq \omega^T.O$ and

\[
\forall \zeta, \zeta \sim_n \omega \Rightarrow \chi \preceq \zeta^T.O \tag{18}
\]

Suppose now $\vartheta \in S \sqcup \Env_{1,0}$ satisfies $\theta \preceq \vartheta^T.I$. (This is the assumption from the causality property (16).) To finish the proof, we need to verify that $\chi \preceq \vartheta^T.O$. We will do it by exhibiting a behavior $\zeta \in S \sqcup \Env_{1,0}$ such that

\[
\zeta^T.I = \vartheta^T.I \quad \text{and} \quad \zeta \sim_n \omega. \tag{19}
\]

In view of (15) and (18), these two properties indeed imply $\chi \preceq \vartheta^T.O$.

We will proceed to define a behavior $\zeta$ satisfying (19). (The definition will take more effort than the proof that it satisfies (19).)
Consider the variation $P^{\alpha,k,n}_Z$ of the producer machine $P^n_Z$, given by

\[
\begin{align*}
P^{\alpha,k,n}_Z &= \{ct \leq n \land tct_Z = m \land m < k \Rightarrow \text{valid}_Z \land Z = \alpha[m]\} \\
\overline{P^{\alpha,k,n}_Z} &= \{ct \leq n \land tct_Z \geq k \Rightarrow \neg \text{valid}_Z \land Z = \text{Arb}_Z\} \\
P^{\alpha,k,n}_Z &= \{ct > n \land tct_Z = m \Rightarrow \text{valid}_Z \land Z = \alpha[m]\}
\end{align*}
\]

where the variable ct denotes the position in the stream, i.e., for every $\sigma \in S$, we set $\sigma[n] = ct = n$. The numbers $k, n$ are arbitrary in this definition, although we will use the producers just defined with the values for $k$ and $n$ as defined in previous paragraphs. The sequence $\alpha$ is arbitrary but infinite. Intuitively, $P^{\alpha,k,n}_Z$ eagerly offers the first $k$ elements of $\alpha$ for transfer while ct $\leq$ n. If all $k$ get transferred then the machine waits until ct $>$ n and then eagerly offers the rest of $\alpha$ for transfer.

It is easy to see that $P^{\alpha,k,n}_Z$ is a subsystem of $P_Z$ and that it is an elastic machine with its two outputs, data$_Z$, valid$_Z$ depending sequentially on its input, stop$_Z$. Moreover, for every behavior $\sigma$ of this machine, one has $\sigma^T.Z \leq \alpha$. In particular, $\sigma^T.Z = \alpha$ for every behavior $\sigma$ such that $\sigma^T$ is infinite.

Define the system

\[
S^\alpha = S \cup P^\alpha \cup \bigcup_{Y \in O} C_Y^\alpha,
\]

Arguing as in the proof of Lemma 16, we obtain that $S^\alpha$ has exactly one behavior $\zeta$ and that it satisfies $\zeta^T.I = \phi^T.I$.

Finally, to deduce $\zeta \sim_n \omega$, note that $P^\alpha$ and $P^\beta$ are machines with identical behaviors "up to $ct = n$"; that is, for every behavior of $P^\alpha$ there is a $\sim_n$-equivalent behavior of $P^\beta$ and vice versa. Therefore, the systems $S^\alpha$ and $S^\beta$ must be in the same relationship. Since $\zeta, \omega$ are the only behaviors of $S^\alpha, S^\beta$ respectively, it follows that $\zeta \sim_n \omega$.

I. Example: Strong Liveness Not Preserved by Feedback

The forward and backward strong liveness conditions for an $[I, O]$-system are given by

\[
\begin{align*}
S & \models (\min_{tct_I} > tct_Y \Rightarrow F \text{valid}_Y) \quad (20) \\
S & \models (\min_{tct_O} \leq tct_X \Rightarrow F \neg \text{stop}_X) \quad (21)
\end{align*}
\]

for every $X \in I$ and for every $Y \in O$.

In practice, elastic components satisfy these strong liveness conditions. However, a composition of such components, while satisfying weaker liveness conditions (2) and (3), may not satisfy the stronger versions (20) and (21). The fundamental difference between the liveness and the strong liveness is as follows: while the strong liveness promises progress on all under-served output (20) or input (21) channels, the liveness only guarantees progress on the least served, so to speak the "hungriest", channels.

Figure 7 shows an example demonstrating that feedback does not preserve the strong forward liveness. Figure 7(a) depicts an elastic machine $S$ with a sequential channel pair $(b, e)$ during a cycle when channel $a$ has seen 2 transfers, channel $d$ - 0 transfers, and all other channels - 1 transfer each. For machine $S$: $\min_{tct_I} = \min_{tct_{\{a,b\}}} = 1 > tct_d$,

![Fig. 7](image-url)

Fig. 7. Example demonstrating that feedback does not preserve the strong forward liveness.

hence (20) implies future progress for $d$: $F \text{valid}_d$. No progress on $e$ can be guaranteed, since $\min_{tct_{\{a,b\}}} = \min_{tct_{\{b,e\}}} = 2 > tct_e$. Progress on $d$ follows immediately from progress on $d$ in machine $S$. Progress on $e$ does not. However, if we can show that the second transfer would eventually occur on the feedback channel $(b, e)$, it would imply progress on $c$. The former holds due to the sequentiality of the input-output channel pair $(b, e)$. The latter does not hold since the strong backward liveness condition for $S$ cannot be applied until the first transfer on the output channel $d$ occurs. Hence (20) does not hold for $F(S)$. Note that (2) is satisfied since it only requires progress on the output channel $d$.

Figure 8 shows an example demonstrating that feedback does not preserve the strong backward liveness either. The counter-example for the strong backward liveness is (almost) symmetric to the forward case. Figure 8(a) depicts an elastic machine $S$ with a sequential channel pair $(b, e)$ during a cycle when channel $a$ has seen 2 transfers, channel $d$ (an input channel in this example) - 0 transfers, and all other channels - 1 transfer each. For machine $S$:

\[
\min_{tct_O} = \min_{tct_{\{c,e\}}} = 1 \geq tct_d = 0
\]

hence (21) implies future progress for input $d$: $F \neg \text{stop}_d$. No progress on $a$ can be guaranteed, since $\min_{tct_{\{c,e\}}} < tct_a$.

To prove the strong backward liveness for $F(S)$, note that $F(S)$ has not seen any input (20) and $F(S)$ channel $b$ is not an input anymore and $\min_{tct_{\{c,e\}}} = 2 > tct_e$. Progress on $d$ follows immediately from progress on $d$ in machine $S$. Progress on $e$ does not. However, if we can show that the second transfer would eventually occur on the feedback channel $(b, e)$, it would imply progress on $c$. The former holds due to the sequentiality of the input-output channel pair $(b, e)$. The latter does not hold since the strong backward liveness condition for $S$ cannot be applied until the first transfer on the output channel $d$ occurs. Hence (20) does not hold for $F(S)$. Note that (2) is satisfied since it only requires progress on the output channel $d$.

![Fig. 8](image-url)

Fig. 8. Example demonstrating that feedback does not preserve the strong backward liveness.

progress on $d$ follows immediately from progress on $d$ in machine $S$. If we can see that the second transfer would eventually occur on the feedback channel $(b, e)$, it would imply
progress on \( a \). By the definition of transfer and the Handshake Lemma 11 it is necessary and sufficient to show that \( F \) valid and \( \neg stop_b \). The latter holds due to the strong backward liveness (21) of the original machine \( S \). The former, however, does not hold since the sequentiability of channel \((b, e)\) cannot be applied until the first transfer on the input channel \( d \) occurs. Hence (21) does not hold for \( F(S) \), while (3) holds since it only requires progress on the input channel \( d \).

**J. Proof of Theorem 3 (a)**

This proof will use results and arguments from the proof of Theorem 2 above. Let us begin with the appropriate strengthening of Lemma 12.

**Lemma 17:** Let 

\[
\text{Env}^k_{I, O, P} = \mathcal{P}^{k-1}_P \cup \bigcup_{X \in I - P} \mathcal{P}^k_X \cup \bigcup_{Y \in O} C_Y
\]

and let \( S \) be an \([I, O]\)-system satisfying the conditions (1-3) and (7). Then for every behavior \( \omega \in S \cup \text{Env}^k_{I, O, P} \) one has \(|\omega^T.Q| \geq k\).

**Proof.** Let \( \omega \in S \cup \text{Env}^k_{I, O, P} \). Clearly, \( \text{Env}^k_{I, O, P} \subseteq \text{Env}^{k-1}_{I, O, P} \), so by Lemma 12, \( \omega \models F \text{tct}_Z \geq k - 1 \) holds for all \( Z \in I \cup O \). Assume \(|\omega^T.Q| < k\), the contrary of what we are to prove. Thus,

\[
\omega \models F (\min_{tct} \geq k - 1 \land \text{tct}_Q = k - 1).
\]  

(22)

We claim now that \( \omega \models F \text{tct}_X \geq k \) for every \( X \in I - P \). To prove this claim, assume it is not true. Then we have \( \omega \models F (\min_{tct} \geq k - 1 \land \text{tct}_X = k - 1) \) for some \( X \in I - P \). Thus, \( \omega \models F \min_{tct} \geq k - 1 \land \text{tct}_X = k - 1 \) holds for all \( X \in I - P \). On the other hand, since \( \omega \models X.\text{valid}_X \land \text{stop}_X \in \mathcal{P}_X \), from \( \omega \models F \text{tct}_X = k - 1 \) and (14) we can derive \( \omega \models G F \text{valid}_X \). Since \( \mathcal{P}_X \) satisfies the persistence condition (1), Lemma 11 implies \( \omega \models G F \text{transfer}_X \), which contradicts our assumption \( \omega \models F \text{tct}_X = k - 1 \). This finishes the proof of the claim.

From (22), the claim above, and the elastic sequentiability condition (7), we obtain \( \omega \models G F \text{valid}_Q \). On the other hand, we have \( \omega \models G \neg \text{stop}_Q \) since \( C_Q \) is a component of our system. Invoking Lemma 11 again, we obtain \( \omega \models G F \text{transfer}_Q \), which contradicts our starting assumption \(|\omega^T.Q| < k\) and so finishes the proof.

Suppose now \( \theta \) is a partial \( I \)-behavior with \(|\theta \cdot P| = k - 1 \) and \(|\theta \cdot X| = k \) for all \( X \in I - P \). Define the system \( S^0 \) in the eager environment by the same equation (17) used in the proof of Theorem 2. In the same way that we derived Lemma 16 from Lemma 12, Lemma 14, and Lemma 13, we can now use Lemma 17 in place of Lemma 12 and prove the following: The system \( S^0 \) has a unique behavior \( \omega \), and that behavior satisfies the properties \( \omega^T.I = \theta \) and \(|\omega^T.Q| \geq k\).

Let \( \chi \) be the prefix of length \( k \) of \( \omega^T.Q \). For the proof that the pair \((P, Q)\) is sequential for \( S^0 \), it suffices to prove 

\[\forall \theta \in S \cup \text{Env}_{I, O} \land \theta \models \omega^T.I \Rightarrow \chi \models \omega^T.Q.\]

This can indeed be derived in the same manner we derived the similar condition (16) in the proof of Theorem 2. For the number \( n \) we choose now any integer such that \( \omega^[[n]] = \text{tct}_Q = k \), and for any \( \vartheta \) such that \( \vartheta \models \omega^T.I \), we define 

\[S^0 = S \cup \mathcal{P}^0 \cup \bigcup_{Y \in O} C_Y,\]

where 

\[\mathcal{P}^0 = \mathcal{P}^0_{P} \cup \bigcup_{X \in I - P} \mathcal{P}^0_{X}.\]

The remaining details of the derivation of \( \chi \models \omega^T.Q \) are now pure repetition of what is done in the proof of Theorem 2 and are therefore omitted.

**K. Proof of Theorem 3 (b)**

We will prove here that \( \langle S \mid P = Q \rangle \) satisfies the persistence and liveness conditions. The transfer determinism (even functionality) property follows from part (c), proved in Section L below.

Let us start with notation:

\[S^\prime = \langle S \mid P = Q \rangle \quad I^\prime = I - \{P\} \quad O^\prime = O - \{Q\}\]

\[\mu_Z = \min_{tct} \geq \text{tct}_Z \quad \mu'_{Z} = \min_{tct} \cup \text{tct}_Z\]

Recall that for every behavior \( \omega \) of \( S^\prime \) there exists a unique behavior \( \omega \) of \( S \) such that

\[\omega \models I' \cup O' = \omega \quad \omega \models \mu_Z \quad \omega \models \mu'_{Z} \quad \omega \models \text{stop}_P \]

(23)

Just from the observation that for each \( Z \in I \cup O \) the stream triples \((\omega.Z, \omega', \text{valid}_Z, \omega', \text{stop}_Z)\) and \((\omega.Z, \omega, \text{valid}_Z, \omega. \text{stop}_Z)\) are equal, it follows that \( S^\prime \) satisfies the persistence conditions (1).

Turning to the liveness conditions, we first prove an auxiliary result.

**Lemma 18:** Let \( \phi_n = \min_{tct} > n \land \min_{tct} \geq n \). Then for every \( n \geq 0 \) and every behavior \( \omega \in S \) as in (23), one has \( \omega \models G (\phi_n \Rightarrow F \text{tct}_P > n) \).

**Proof.** The proof is by induction on \( n \). We can assume 

\[\omega \models G (\phi_n \Rightarrow F \text{tct}_P \geq n).\]

(24)

Indeed, this is trivially true for \( n = 0 \), and for \( n > 0 \) it follows from the induction hypothesis \( \omega \models G (\phi_{n-1} \Rightarrow F \text{tct}_P > n - 1) \) and the obvious fact \( \omega \models \phi_n \Rightarrow \phi_{n-1} \).

Arguing by contradiction, we can assume 

\[\omega \models F (\phi_n \land \text{tct}_P \leq n),\]

which, in view of (24), can be sharpened into 

\[\omega \models F (\phi_n \land \text{tct}_P = n).\]

Since \( \omega \models G (\phi_n \Rightarrow G \phi_n) \) (by monotonicity of \( \text{tct}_Z \)), this implies 

\[\omega \models F G (\phi_n \land \text{tct}_P = n).\]

(25)

Clearly, 

\[\omega \models G (\phi_n \land \text{tct}_P = n \Rightarrow \mu_P),\]

which with (25) gives \( \omega \models F G \mu_P \), which in turn, combined with (3), gives 

\[\omega \models F G \neg \text{stop}_P.\]

(26)
Since $\omega \models G(t_{ct}^P = t_{ct}^Q)$, we have

$$\omega \models G(\phi_n \land t_{ct}^P = n \Rightarrow \mu_Q \land \min_{t_{ct}^P} > t_{ct}^Q),$$

which with (25) gives $\omega \models F G (\mu_Q \land \min_{t_{ct}^P} > t_{ct}^Q)$, which in turn, combined with (7) gives

$$\omega \models F G F \text{valid}_Q.$$  \hfill (27)

From (26) and (27), we derive $\omega \models (G F \neg \text{stop}_P) \land (G F \text{valid}_P)$. Then by Lemma 11, we have $\omega \models G F \text{transfer}_P$, which implies $\omega \models F t_{ct}^P > n$, contradicting (25). This finishes the proof of Lemma 18.

The liveness conditions for $S'$ amount to the following, to be proved for every $\omega$ as in (23).

$$\omega \models G((\mu'_Y \land \min_{t_{ct}^P} > t_{ct}^Y) \Rightarrow F \text{valid}_Y) \quad (28)$$

$$\omega \models G(\mu'_X \Rightarrow F \neg \text{stop}_X) \quad (29)$$

for every $X \in I'$ and every $Y \in O'$.

We prove the simpler property (29) first. Relying on the backward liveness of $S$ (3), it suffices to check

$$\omega \models G((\mu'_X \Rightarrow F \neg \text{stop}_X) \lor F \mu_X),$$

which is equivalent to

$$\omega \models G(\mu'_X \land G \text{stop}_X \Rightarrow F \mu_X).$$

It is easy to verify that

$$\omega \models G(G \text{stop}_X \Rightarrow \exists n. G t_{ct} X = n),$$

so it suffices to prove

$$\omega \models G(\mu'_X \land G t_{ct} X = n \Rightarrow F \mu_X).$$

Since $\omega \models G(\mu'_X \land t_{ct} X = n \Rightarrow G \mu'_X)$, our goal can be restated as

$$\omega \models G(\mu'_X \land G t_{ct} X = n \Rightarrow F(\mu'_X \land t_{ct} X \geq n))$$

and then, in view of $\omega \models G(\mu'_X \land G t_{ct} X = n \Rightarrow G \mu'_X)$, reduced to

$$\omega \models G(\mu'_X \land G t_{ct} X = n \Rightarrow F t_{ct}^P \geq n).$$

Since $\omega \models G(\mu'_X \land t_{ct} X = n \Rightarrow \phi_{n-1})$, Lemma 18 finishes the proof.

For the proof of (28), notice first that, in view of (2), it suffices to prove

$$\omega \models G((\mu'_Y \land \psi \Rightarrow F \text{valid}_Y) \lor F(\mu'_Y \land t_{ct}^P > t_{ct}^Y \land \psi)),$$

where $\psi = \min_{t_{ct}^P} > t_{ct}^Y$. This is equivalent to

$$\omega \models G((\mu'_Y \land \psi \land G \neg \text{valid}_Y) \Rightarrow F(\mu'_Y \land t_{ct}^P > t_{ct}^Y \land \psi)).$$

Since

$$\omega \models G(G \neg \text{valid}_Y \Rightarrow \exists n. G t_{ct} Y = n),$$

it suffices to prove

$$\omega \models G(\mu'_Y \land \psi \land G t_{ct} Y = n \Rightarrow F(\mu'_Y \land t_{ct}^P > t_{ct}^Y \land \psi)).$$

Since $\omega \models G(\mu'_Y \land t_{ct}^P > t_{ct}^Y \Rightarrow \mu_Y)$, we can restate this goal as

$$\omega \models G(G t_{ct} Y = n \Rightarrow \mu'_Y \land \psi \Rightarrow F(\mu'_Y \land \psi \land t_{ct}^P > n)).$$

Now we use $\omega \models G(G t_{ct} Y = n \Rightarrow G(\mu'_Y \land \psi \Leftrightarrow \phi_n))$ to further rewrite our goal as

$$\omega \models G(G t_{ct} Y = n \Rightarrow \phi_n \Rightarrow F(\phi_n \land t_{ct}^P > n)).$$

This finally follows from the already seen $\omega \models G(\phi_n \Rightarrow \phi_n)$ (true by monotonicity of tct) and Lemma 18.

L. Proof of Theorem 3 (c)

We continue with the notation $I', O', S'$ from the previous section.

We need to prove that for every $\omega' \in S' \cup \text{Env}_{I',O'}$ there exists $\omega \in S \cup \text{Env}_{I,O}$ such that $\omega^\tau(I' \cup O') = \omega'^\tau$ and $\omega^\tau.\bar{P} = \omega'^\tau.\bar{Q}$. Indeed, we have already seen that, given $\omega' \in S'$, there exists $\omega \in S$ such that

$$\omega.(I' \cup O') = \omega' \quad \omega.\bar{P} = \omega.\bar{Q} \quad \omega.\text{valid}_P = \omega.\text{valid}_Q \quad \omega.\text{stop}_P = \omega.\text{stop}_Q$$

(see (23) above), so it only remains to prove that $\omega \in \text{Env}_{I,O}$ holds for this $\omega$.

By Part (b) of the theorem (proved in the previous section) and Lemma 3 we know that $\omega^\tau.\bar{S}$ is infinite for every $\bar{S} \in I' \cup O'$. It follows then from Lemma 18 that the sequences $\omega^\tau.\bar{P}$ and $\omega^\tau.\bar{Q}$ are (equal and) infinite. This implies that $\omega.\{P,\text{valid}_P,\text{stop}_P\} \in \bar{P}$ and $\omega.\{Q,\text{valid}_Q,\text{stop}_Q\} \in \bar{Q}$. (The persistence condition in the definition of $\bar{P}$ is equivalent to the persistence condition for the wire $Q$ for $\omega$ and is satisfied because $\omega \in S$ and $Q$ is an output of $S$.) Since $\text{Env}_{I',O'} = \text{Env}_{I,O} \cup \bar{P} \cup \bar{Q}$ and $\omega' \in \text{Env}_{I',O'}$, we finally obtain the desired $\omega \in \text{Env}_{I,O}$. 

M. Sequentiality Interface for Elastic Feedback

Definition 16: Given a function $\delta: O \rightarrow 2^I$ and $M \in I, N \in O$, the function $\delta_{[M=N]}: O \rightarrow \{N\} \rightarrow 2^{\{M\}}$ is defined by

$$\delta_{[M=N]}(Y) = \begin{cases} \delta(Y) - \{M\} & \text{if } M \in \delta(Y) \\ \{\delta(Y) \cap N \} \setminus \delta(N) & \text{otherwise} \end{cases}$$

The following formula is another characterization of $\delta_{[M=N]}$:

Lemma 19: For every $X \in I - \{M\}$ and $Y \in O - \{N\}$, $X \notin \delta_{[M=N]}(Y)$ iff $X \notin \delta(Y) \lor (X \notin \delta(N) \land M \notin \delta(Y))$.

Proof. Easy.

Definition 17: The persistent update $\delta' = (\delta_{[P=Q]})_{[M=N]}$ has $X \notin \delta'(Y)$ if and only if

$$X \sim Y \lor (X \sim X N \land \sim M \sim Y) \lor (X \sim Q \land P \sim Y)$$

$$\lor (X \sim Q \land P \sim N \land M \sim Y),$$

$$\lor (X \sim Q \land P \sim N \land M \sim Y).$$
where, to avoid clutter, we used the notation $X \rightarrow Y$ for $X \notin \delta(Y)$.

**Lemma 21**: If $\delta$ is a sequentiability interface for an elastic machine $\mathcal{S}$ and if $M \in \delta(N)$, then $\delta_{[M=N]}$ is a sequentiability interface for $<\mathcal{S} [ M = N ]>$.

**Proof.** Let us start again with notation:

$$S' = \langle S [ M = N ] \rangle$$

$$I' = I - \{ M \}$$

$$O' = O - \{ N \}$$

$$\mu'_{Z} = \min_{\text{tct}_{I' \cup O'}} \geq \text{tct}_{Z}$$

For every $Q \in O'$ we need to prove

$$\omega \models G(\mu'_{Q} \land \land \text{tct}_{I' \delta'(Q)} > \text{tct}_{Q} \land F \text{valid}_{Q}),$$

where $\omega$ is a behavior of $\mathcal{S}$ that corresponds to a behavior of $S'$, i.e.

$$\omega.M = \omega.N \land \omega.\text{valid}_{M} = \omega.\text{valid}_{N} \land \omega.\text{stop}_{M} = \omega.\text{stop}_{N}$$

Consider first the case when $M \in \delta(Q)$. By definition of $\delta'$, we have $I' - \delta'(Q) = I - \delta(Q)$ and accordingly restate our goal:

$$\omega \models G(\mu'_{Q} \land \land \text{tct}_{I - \delta(Q)} > \text{tct}_{Q} \land F \text{valid}_{Q}).$$

With the shorthand notation $\psi = \min_{\text{tct}_{I - \delta(Q)} > \text{tct}_{Q}}$, let us assume the contrary:

$$\omega \models F(\mu'_{Q} \land \land \psi \land G \neg \text{valid}_{Q}).$$

Since $\omega \models G(\neg \text{valid}_{Q} \land \exists n. G \land \text{tct}_{Q} > n)$, we have

$$\omega \models F(\mu'_{Q} \land \land \psi \land G \land \text{tct}_{Q} = n)$$

for some $n$, and therefore

$$\omega \models F \land G(\mu'_{Q} \land \land \psi \land G \land \text{tct}_{Q} = n).$$

(30)

Note now that

$$\sigma \models G(\mu'_{Q} \land \land \text{tct}_{Q} = n \Rightarrow \phi_{n-1}),$$

where $\phi$ is as in Lemma 18. Lemma 18 is true in the current context, so we have

$$\omega \models G(\mu'_{Q} \land \land \psi \land \text{tct}_{Q} = n \Rightarrow \text{valid}_{Q} \land \text{tct}_{M} \geq n).$$

(31)

From (31) and (32), we derive

$$\omega \models G(\mu'_{Q} \land \land \psi \land \text{tct}_{Q} = n \land \text{tct}_{M} \geq n).$$

This implies

$$\omega \models F \land G(\mu'_{Q} \land \land \psi),$$

and then, in view of (8),

$$\omega \models F \land G(\mu'_{Q} \land \land \psi \land \text{valid}_{Q}),$$

which directly contradicts (30), finishing the first branch of the proof.

Suppose now $M \notin \delta(Q)$. By definition of $\delta'$, we have $I' - \delta'(Q) = I' - \delta(Q) \cap \delta(N)$ and our goal is

$$\omega \models G(\mu'_{Q} \land \land \text{tct}_{I' \delta'(Q)} > \text{tct}_{Q} \land F \text{valid}_{Q}).$$

(33)

Assuming the contrary, we obtain, in a similar fashion as before,

$$\omega \models F \land G(\theta \land \text{tct}_{Q} = n \land \theta \land \text{valid}_{Q}),$$

for some $n$, where $\theta$ stands for the antecedent part of the implication in (33). Again by Lemma 18, we have

$$\omega \models G(\theta \land \text{tct}_{Q} = n \Rightarrow \text{tct}_{M} \geq n),$$

which, combined with the previous formula, implies

$$\omega \models F \land G(\theta \land \text{tct}_{Q} = n \land \lnot \text{valid}_{Q} \land \lnot \text{tct}_{M} \geq n).$$

This further implies (by monotonicity of $\text{tct}_{M}$)

$$\omega \models F \land G(\theta \land \text{tct}_{Q} = n \land \lnot \text{valid}_{Q} \land \text{tct}_{M} \geq n).$$

(35)

Now, it is clear that one of the following must hold:

$$\omega \models F(\theta \land \text{tct}_{Q} = n \land \lnot \text{valid}_{Q} \land \text{tct}_{M} < n),$$

(34)

$$\omega \models F(\theta \land \text{tct}_{Q} = n \land \lnot \text{valid}_{Q} \land \text{tct}_{M} = n).$$

(35)

We proceed to prove that both (34) and (35) lead to contradiction.

Suppose (34) is true. Notice that

$$\omega \models F(\mu'_{Q} \land \land \text{tct}_{M} > n \land \text{tct}_{Q} = n \Rightarrow \mu'_{Q} \land \land \text{tct}_{I' \delta(Q)} > \text{tct}_{Q}),$$

so by combining with (8) we obtain

$$\omega \models F(\mu'_{Q} \land \land \text{tct}_{M} > n \land \text{tct}_{Q} = n \Rightarrow \text{valid}_{Q}),$$

which (since $\mu'_{Q}$ is a conjunct of $\theta$) contradicts (34).

Suppose (35) is true. Since $\omega \models \text{tct}_{M} = \text{tct}_{N}$, we have

$$\omega \models G(\theta \land \text{tct}_{Q} = n \land \theta \land \text{valid}_{Q} \land \theta \land \text{tct}_{M} \geq n) \quad \text{.}$$

Since $M \in \delta(N)$, we have $I - \delta(N) \subseteq I' - \delta(Q) \cap \delta(N)$. With these observations, we derive

$$\omega \models G(\theta \land \text{tct}_{Q} = n \land \theta \land \text{valid}_{Q} \land \theta \land \text{tct}_{M} \geq n).$$

In view of (8), this implies

$$\omega \models G(\theta \land \text{tct}_{Q} = n \land \text{valid}_{Q}).$$

(36)

Since $\omega \models \text{tct}_{M} = \text{tct}_{N}$ and since $\theta$ has $\mu'_{N}$ as one of the conjuncts, by liveness of $\mathcal{S}$ (3) we also get

$$\omega \models G(\theta \land \text{tct}_{Q} = n \land \theta \land \text{valid}_{Q} \land \theta \land \text{tct}_{M} \geq n) \quad \text{.}$$

Since $\omega \models \text{valid}_{N} = \text{valid}_{M}$, from (36), (37), and (35) we get

$$\omega \models F \land G(\text{valid}_{N} \land F \land \text{valid}_{M} \land \text{valid}_{Q}).$$

(38)

By Lemma 11 (and $\omega \models \text{valid}_{M} = \text{valid}_{N}$), we obtain $\omega \models F \land \text{tct}_{M} > n$, contradicting (35). 

$\Box$
N. Proof of Theorem 4

In Section D, we disambiguated the use of the dependency symbol $\Delta$ and now we do the same for $\Delta'$. 

Definition 17: The dependency graph $D^e(S, \delta)$ of an elastic $[I, O]$-machine $S$ with a sequiency interface $\delta$ has the vertex set $I \cup O$ and a directed edge $(X, Y)$ for every $X, Y$ such that $X \not\in \delta(Y)$. The dependency graph $\Delta^e(N)$ of an elastic network $N = \langle S_1, \ldots, S_m \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle$ is $(D^e(S_1, \delta_1) \sqcup \cdots \sqcup D^e(S_m, \delta_m))[x_1 = y_1, \ldots, x_n = y_n].$

Following the analogy with ordinary networks, we will write $N \sim N'$ when two elastic network determine the same elastic system.

Lemma 22: Let $S = S_1 \sqcup \cdots \sqcup S_m$, where each $S_i$ is an $[I_i, O_i]$-elastic machine with a sequiency interface $\delta_i$. Let $I = \bigcup_{i=1}^m I_i$ and $O = \bigcup_{i=1}^m O_i$, and assume that these unions are disjoint. Define $\delta : O \rightarrow 2^I$ by $\delta(Y) = \delta_i(Y) \sqcup \bigcup_{j \neq i} I_j$ for $Y \in O$. Then $S$ is an $[I, O]$-machine, $\delta$ is a sequiency interface for it, and $D^e(S, \delta)$ is the direct sum of graphs $D^e(S_i, \delta_i).

Proof. Straightforward checking.

Lemma 23: Under the assumptions of Lemma 22, $S^* = S_1^* \sqcup \cdots \sqcup S_m^*$.

Proof. Straightforward checking.

Definition 18: Given an elastic network $N = \langle S_1, \ldots, S_m \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle$ of elastic machines with disjoint inputs and outputs, let $\delta$ be as in Lemma 22, and define $\delta_N = \delta_i[x_1 = y_1, \ldots, x_n = y_n].$

Remark. By Lemma 20, the order of the $n$ operations that lead from $\delta$ to $\delta_N$ is irrelevant.

Lemma 24: If $\delta$ is a sequiency interface of $S$ and $X \notin \delta(Y)$, then $D^e(S[X = X', \delta[x = y])] \hookrightarrow D^e(S, \delta)[u = v]$.

Proof. (Mutatis mutandis, the proof is the same as that of Lemma 10.) The lemma follows from $D^e(S[X = Y]) \hookrightarrow D^e(S)[X = Y]$, where the second immersion is an instance of Lemma 9. For the first immersion, we prove that $D^e((S[X = Y]))$ is actually a subgraph of $D^e(S[X = Y]).$ Notice that these graphs have a common set of vertices $I \cup O - \{X, Y\}. By Lemma 19,

$W \notin \delta_i[x = y](Z)$ iff $W \notin \delta(Z) \lor (W \notin \delta(Y) \land X \notin \delta(Z))$

The left-hand side of this relation says that $(W, Z)$ is an edge of $D^e((S[X = Y]), \delta[x = y])$, while the right-hand side implies that $(W, Z)$ is an edge of $D^e(S, \delta[X = Y]).$

Starting the proof of Theorem 4, consider the following claim.

If $N = \langle S_1, \ldots, S_m \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle$ and both graphs $\Delta(N)$, $\Delta'(N)$ are acyclic, then $N$ is an elastic machine, $\delta_N$ is a sequiency interface for it, and $D^e(N, \delta_N) \hookrightarrow D^e(N').$

We prove this claim by induction on $n$. The case $n = 0$ follows immediately from Lemma 22.

For the induction step, let $N' = \langle S_1, \ldots, S_m \mid X_1 = Y_1, \ldots, X_{n-1} = Y_{n-1} \rangle$. Note that $\Delta^e(N) = \Delta^e(N')[x_n = y_n]$ and $\delta_N = (\delta_N)[x_n = y_n]$. Since $\Delta(N')$ has the acyclic graph $\Delta(N)$ as a quotient, it must be acyclic too. Similarly, $\Delta^e(N')$ must be acyclic as well. By induction hypothesis, $N'$ is an elastic machine with a sequiency interface $\delta_{N'}$, so we can write $N \sim \langle N' \mid X_n = Y_n \rangle$.

We would like to use Theorem 3(b) to conclude that $N$ is an elastic machine, but there are two conditions to check: (1) $(X_n, Y_n)$ is a sequential channel pair for the (elastic system determined by) $N'$, and (2) $\Delta(\langle N' \mid X_n = Y_n \rangle)$ is acyclic.

For (1), it suffices to prove that $X_n \in \delta_{N'}(Y_n)$. If this were not true, there would be a path from $X_n$ to $Y_n$ in $D^e(N', \delta_{N'})$. By induction hypothesis, this graph is immersed in $\Delta^e(N')$ so there would be a path from $X_n$ to $Y_n$ in $\Delta^e(N')$, so $\Delta(N) = \Delta^e(N')[x_n = y_n]$ would not be acyclic, which it is. This contradiction proves (1).

For (2), we have the derivation

$\Delta(\langle N' \mid X_n = Y_n \rangle) \hookrightarrow D(N'(X_n = Y_n), \delta_{N'})[x_n = y_n] 
\hookrightarrow \Delta(N')(X_n = Y_n, \delta_{N'})[x_n = y_n] 
\hookrightarrow \Delta(N'),$

where the immersion holds because the network $N'$ is a machine (induction hypothesis) and we found in the proof of Theorem 1 that $D(M) \hookrightarrow \Delta(M)$ holds for every machine $M$.

We have now completed the proof that $N$ is an elastic machine. To finish the induction step, we need to check that $\delta_N$ is a sequiency interface for $N$ and that $D^e(N, \delta_N) \hookrightarrow \Delta^e(N)$.

The first of these statements follows directly from Lemma 21 and the fact $X_n \in \delta_N(X_n)$ (proved above). For the second, we have the following derivation:

$D^e(N', \delta_{N'}) \hookrightarrow D^e(\langle N' \mid X_n = Y_n \rangle, \delta_{N'})[x_n = y_n] 
\hookrightarrow \Delta^e(N')(X_n = Y_n) 
\hookrightarrow \Delta^e(N).$

The first immersion here is justified by Lemma 24 and the second is justified by the induction hypothesis together with Lemma 8.

We have finished the inductive proof of the claim that generalizes part of Theorem 4. It remains to prove that if $N = \langle S_1, \ldots, S_m \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle$ and both graphs $\Delta(N)$, $\Delta'(N)$ are acyclic, then $N = \langle S_1^*, \ldots, S_m^* \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle$ is a machine, and $N^* = N$.

Observe that Theorem 3(a) implies $D(S) \hookrightarrow D'(S, \delta)$ for every elastic machine $S$ and a sequiency interface $\delta$ for it. As a consequence,

$\Delta(N) \hookrightarrow \Delta'(N) \hookrightarrow \Delta(N),$

Since $\Delta(N)$ is acyclic, it follows that $\Delta(N)$ is acyclic too, and then by Theorem 1 that $N$ is a machine.
Finally, the equality $\mathcal{N}^\tau = \mathcal{N}$ is proved by induction on $n$. The case $n = 0$ is Lemma 23. For the induction step, we have

$$
\begin{align*}
\mathcal{N}^\tau & = (\langle \mathcal{N}' | X_n = Y_n \rangle)^T \\
& = (\langle \mathcal{N}' | X_n = Y_n \rangle) \\
& = (\mathcal{N}' | X_n = Y_n) \\
& = \mathcal{N},
\end{align*}
$$

where $\mathcal{N}' = \langle S_1^\top, \ldots, S_m^\top | X_1 = Y_1, \ldots, X_{n-1} = Y_{n-1} \rangle$. The second equality here follows from Theorem 3(b), and the third follows by induction hypothesis.

### O. Proof of Theorem 5

We begin with an easy lemma about ordinary machines.

Lemma 25: If $\mathcal{S}$ is a machine with a sequential pair $(X_1, Y_1)$ then

$$
\begin{align*}
\langle \mathcal{S} | X_1 = Y_1 \rangle & = \{ \mathcal{S} \sqcup \text{Conn}(X, Y) | X = Y_1, X_1 = Y_1 \}.
\end{align*}
$$

Let $\mathcal{N}' = \langle S_1, \ldots, S_m | | X_2 = Y_2, \ldots, X_n = Y_n \rangle$. We have

$$
\begin{align*}
\Delta^e(\mathcal{N}) & = \Delta^e(\mathcal{N}')|_{X_1 = Y_1} \\
\Delta^e(\mathcal{M}) & = (\Delta^e(\mathcal{N}') \sqcup \Delta^e(\mathcal{B}, \delta_0))|_{X = Y_1, X_1 = Y_1}
\end{align*}
$$

where $\delta_0$ is a sequentiality interface for $\mathcal{B}$, which must be given either by $\delta_0(Y) = \{ X \}$ or $\delta_0(Y) = \emptyset$. A simple analysis shows that the existence of a cycle in $\Delta^e(\mathcal{M})$ implies the existence of a cycle in $\Delta^e(\mathcal{N})$ (Figure 9).

Theorem 4 now implies

$$
\begin{align*}
\mathcal{N}^\top & = \langle (\mathcal{N}')^\top | X_1 = Y_1 \rangle \\
\mathcal{M}^\top & = \langle (\mathcal{N}')^\top \sqcup \mathcal{B}^\top | X = Y_1, X_1 = Y_1 \rangle
\end{align*}
$$

Since $\mathcal{B}^\top = \text{Conn}(X, Y)$, Lemma 25 completes the proof of Theorem 5.