Concurrent Transition System Semantics of Process Networks

Eugene W. Stark
State University of New York at Stony Brook
Stony Brook, NY 11794

Abstract

Using *concurrent transition systems* [Sta86], we establish connections between three models of concurrent process networks, *Kahn functions*, *input/output automata*, and *labeled processes*. For each model, we define three kinds of algebraic operations on processes: the *product* operation, *abstraction* operations, and *connection* operations. We obtain homomorphic mappings, from input/output automata to labeled processes, and from a subalgebra (called "input/output processes") of labeled processes to Kahn functions. The proof that the latter mapping preserves connection operations amounts to a new proof of the "Kahn Principle." Our approach yields: (1) extremely simple definitions of the process operations; (2) a simple and natural proof of the Kahn Principle that does not require the use of "strategies" or "scheduling arguments"; (3) a semantic characterization of a large class of labeled processes for which the Kahn Principle is valid, (4) a convenient operational semantics for nondeterminate process networks.

1 Introduction

A "dataflow network" consists of a collection of concurrently executing processes that communicate data values over channels. In Kahn's model [Kah74,KM77,Mac79] for such networks, a process with m-input and n-output channels is represented by a continuous function $\phi : H^m \to H^n$, where H is a suitable cpo of "channel histories." The theory of continuous functions on cpo's permits elegant definitions to be given for various kinds of operations on processes. In particular, the operation of composing processes into a network can be defined in terms of least fixed points of continuous functionals. The relationship between process composition and least fixed point has been referred to [Par82,Fau82] as the "Kahn Principle."

Kahn's model is highly satisfactory as a semantic model for dataflow networks, except in one respect: it is insufficiently general in the sense that only "determinate" processes (those with functional input/output behavior) can be represented. However, nondeterminate processes such as "merge," which merges streams of data values arriving on two input channels into a single output stream, are also interesting and important in applications. Attempts to extend Kahn's model to the nondeterminate case, *e.g.* [Kel78,BA81,Par82,Pra82,Fau82,BM82,Bro83,KP84], [SN85,Kok86], have not been entirely successful.

To satisfactorily resolve the problem of semantics of nondeterminate process networks, not one, but *two* mathematical models, an *operational* and a *denotational* model, are necessary, along with a mapping that takes each object of the operational model to a corresponding element of the denotational model, in such a way that the relevant operations on processes are preserved. If such a mapping exists, then we may say that the denotational model is *correct* with respect to the operational semantics. Often researchers have constructed denotational models and performed substantial investigations of their properties, without showing their model to be correct with respect to a suitable operational model. Even Kahn's original paper avoids the issue of proving the correctness of his fixpoint characterization of the behavior of process networks, referring the reader instead to "a similar set up" in work of Cadiou [Cad72].

One possible reason why formal operational models have often not been considered, is that existing such models are mathematically inconvenient, and do not facilitate proofs of relationships with more abstract, denotational models. We would argue that this mathematical inconvenience is due in large part to the problems that arise when interleaved step models of concurrency are combined with a desire to treat infinite computations. This combination leads to a need for "strategies," "scheduling arguments," "fairness predicates," and their ilk, with consequent complications in proofs. For example, Faustini [Fau82] requires the notion of a winning strategy for a two-player infinite game of perfect information to relate his operational model to Kahn's model.

In this paper, we show how an operational model for concurrent process networks can be based on the "concurrent transition systems" of [Sta86]. The convenience and utility of this approach is demonstrated by the following results:

- We are able to give extremely simple definitions of the operations of *product*, *abstraction*, and *connection* on processes. These operations allow various ways of constructing process networks to be modeled.

- We can give a straightforward and natural proof of the Kahn Principle. In particular, this proof does not require any complicated scheduling or fairness arguments.

- We obtain connections between three different models of concurrent process networks: Kahn's model, an *input/output automaton* model with a classical automata-theoretic flavor reminiscent of [AM75,SAM83], and a *labeled process* model which is closely related to the *labeled transition system* models that have been used in the study of CCS and CSP [Mil80,Hoa78,BR83].

- We derive a reasonably simple semantic characterization of a large class of labeled processes for which the Kahn Principle is valid.

- We obtain a convenient operational model of nondeterminate process networks.

The fundamental innovation that makes our approach an improvement over previous work is the use of "concurrent transition systems," which are ordinary transition systems that have been augmented with concurrency information. The concurrency information in a concurrent transition system induces a congruence on the collection of all computation sequences of the underlying ordinary transition system, in which two computation sequences are congruent exactly when they represent two possible sequential interleavings of the same underlying concurrent computation. By factoring the collection of computation sequences by this congruence, we are able to obtain a convenient definition of computations as ideals of a partial order, and to replace the troublesome notion of a "fair computation sequence" with the more tractable notion of a "maximal ideal."

2 Concurrent Transition Systems

In this section we define concurrent transition systems, and summarize their properties, derived in a previous paper [Sta86]. A concurrent transition system (CTS) is an ordinary transition system (*i.e.* a graph), which is equipped with a "translation operation" $\uparrow$ that captures concurrency information, and a collection of "identity arrows" that behave as identities for $\uparrow$ in a certain sense. The objects of the transition system are "states," and the arrows are "transitions." If f and g are two transitions from a single state q, then f and g are either "inconsistent," in which case they represent incompatible nondeterministic choices that can be made from q, or "consistent," in which case f and g represent actions that might appear as part of a single concurrent computation. In case f and g are consistent, then we can form the transitions $f \uparrow g$ (f "after" g) and $g \uparrow f$. Intuitively, $f \uparrow g$ can be thought of as what remains of the transition f after g has occurred. For example, if f and g represent steps of independent processes A and B, respectively, in a system of processes, that can be executed from a global system state q, then $f \uparrow g$ represents the same step of process A as f does, except that $f \uparrow g$ is executed starting from the state that results after g is executed from q.

We now give the formal axioms for a CTS, and sketch the development of their consequences.

A *graph* is a tuple $G = (O, A, \mathrm{dom}, \mathrm{cod})$, where O is a set of *objects*, A is a set of *arrows*, and dom, cod are functions from A to O, which map each arrow to its *domain* and *codomain*, respectively. Arrows f, g of G are called *composable* if $\mathrm{cod}(f) = \mathrm{dom}(g)$ and *coinitial* if $\mathrm{dom}(f) = \mathrm{dom}(g)$. Let $\mathrm{Coin}(G)$ denote the set of all coinitial pairs of arrows of G.

If $G = (O, A, \mathrm{dom}, \mathrm{cod})$ is a graph, then define the *augmented graph* $G^\dagger = (O^\dagger, A^\dagger, \mathrm{dom}^\dagger, \mathrm{cod}^\dagger)$ to be the graph defined by: $O^\dagger = O \cup \{\Omega\}$, $A^\dagger = A \cup \{\omega_q : q \in O^\dagger\}$, $\mathrm{dom}^\dagger(\omega_q) = q$, and $\mathrm{cod}^\dagger(\omega_q) = \Omega$, where $O \cap \{\Omega\} = \emptyset$, and $A \cap \{\omega_q : q \in O^\dagger\} = \emptyset$.

A *concurrent transition system* is a triple $(G^\dagger, \mathrm{id}, \uparrow)$, where

- $G = (O, A, \mathrm{dom}, \mathrm{cod})$ is a graph, called the *underlying graph*. The elements of $O^\dagger$ (resp. O) are called *(proper) states* and the elements of $A^\dagger$ (resp. A) are called *(proper) transitions*.

- $\mathrm{id} : O^\dagger \to A^\dagger$ maps each $q \in O^\dagger$ to a distinguished *identity transition* id_q.

- $\uparrow : \mathrm{Coin}(G^\dagger) \to A^\dagger$ is a function, called the *translation operation*. We write $f \uparrow g$ for $\uparrow (f, g)$.

These data are required to have the following properties:

1. For all $q \in O^\dagger$, and all coinitial $(f, g) \in A^\dagger \times A^\dagger$, $\mathrm{dom}(\mathrm{id}_q) = \mathrm{cod}(\mathrm{id}_q) = q$; $\mathrm{dom}(f \uparrow g) = \mathrm{cod}(g)$; and $\mathrm{cod}(f \uparrow g) = \mathrm{cod}(g \uparrow f)$.

2. For all $f : q \to r$ in $A^\dagger$, $\mathrm{id}_q \uparrow f = \mathrm{id}_r$; $f \uparrow \mathrm{id}_q = f$; $f \uparrow f = \mathrm{id}_r$; $\omega_q \uparrow f = \omega_r$; and $f \uparrow \omega_q = \mathrm{id}_\Omega = \omega_\Omega$.

3. For all coinitial $f, g, h \in A^\dagger$, we have
$$(h \uparrow f) \uparrow (g \uparrow f) = (h \uparrow g) \uparrow (f \uparrow g).$$

4. For all $f, g \in A^\dagger$ with $f : q \to r$ and $g : q \to s$, if $f \uparrow g = \mathrm{id}_s$ and $g \uparrow f = \mathrm{id}_r$, then $f = g$.

Coinitial transitions f, g are called *consistent* if $f \uparrow g \neq \omega_{\mathrm{cod}(g)}$ (equivalently, if $g \uparrow f \neq \omega_{\mathrm{cod}(f)}$). Note that every graph lifts to a CTS in such a way that f, g are consistent iff $f = g$ or one is an identity. A CTS M is *determinate* if every coinitial pair of transitions of M is consistent.

Define the relation $\preceq$ on the transitions of a CTS by: $f \preceq g$ iff f, g are coinitial and $f \uparrow g = \mathrm{id}_{\mathrm{cod}(g)}$. We call $\preceq$ the *prefix relation*. It can be shown that, for each q, the prefix relation partially orders the set $M(q, \cdot)$ of all proper transitions of M with domain q. We say that a transition h is a *join* of the coinitial transitions f, g if $f \preceq h$, $g \preceq h$, $h \uparrow f = g \uparrow f$, and $h \uparrow g = f \uparrow g$. It can be shown that joins coincide with least upper bounds under prefix order, and we write $f \vee g$ to denote the join of f and g when it exists.

Suppose f, g are composable. We say that a transition h is a *composite* of f, g if $f \preceq h$ and $h \uparrow f = g$. It can be shown that a composite of f, g, if it exists, is unique, and we denote it by fg. A CTS is *complete* if every composable pair of transitions has a composite. It can be shown that if either of $f(g \uparrow f)$, $f \vee g$ exists, then the other does, also, and the two are equal. Thus completeness implies join-completeness.

A large number of algebraic laws, relating translation, join, and composition, hold in a complete CTS. In short, these laws can be summarized by saying that a complete CTS is a category [ML71,AM75] with a terminal object (Ω), pushouts (given by $\uparrow$), and in which all arrows are epi and the only isomorphisms are identities. In later sections, the laws summarized below will be used frequently, often without explicit mention.

Proposition 1 *The following laws hold in a complete CTS:*

1. *For all coinitial f, g, h,*
   (a) $h \uparrow (f \vee g) = (h \uparrow f) \uparrow (g \uparrow f)$.
   (b) $(f \vee g) \uparrow h = (f \uparrow h) \vee (g \uparrow h)$.
2. *For all f, g, h, if f, h are coinitial and f, g are composable, then*
   (a) $h \uparrow fg = (h \uparrow f) \uparrow g$.
   (b) $fg \uparrow h = (f \uparrow h)(g \uparrow (h \uparrow f))$.

2.1 Constructions in CTS

Suppose M and M' are CTS's, with underlying graphs G and G', respectively. A *CTS-morphism* from M to M' is a graph homomorphism $F : G^\dagger \to (G')^\dagger$, such that

1. $F(\Omega) = \Omega$, and $F(q) \neq \Omega$ for all proper states q of M.
2. $F(\mathrm{id}_q) = \mathrm{id}'_{F(q)}$ and $F(\omega_q) = \omega_{F(q)}$ for all states q of M.
3. $F(f \uparrow g) = F(f) \uparrow' F(g)$ for all transitions f, g of M.

It is easily verified that identity graph homomorphisms are CTS-morphisms, and that the class of CTS-morphisms is closed under composition. Thus the class of all concurrent transition systems, equipped with their morphisms, forms a category **CTS**. In the sequel, the term "morphism" will mean "CTS morphism" unless otherwise specified.

The category **CTS** can be shown to have an initial and terminal object, all small coproducts, and all small limits. This means that a large number of interesting operations can be defined on **CTS**, using the algebraic language of concurrent transition systems and their morphisms. In this paper, however, we shall need only the *terminal object*, the *product* construction, and the *equalizer* construction.

2.1.1 Terminal Object

Let $\mathbf{1}$ be the CTS whose underlying graph has just one proper state, $\imath$, and one proper transition, $\mathrm{id}_\imath$.

Proposition 2 *The CTS* $\mathbf{1}$ *is a terminal object in the category* **CTS***.*

2.1.2 Product

Suppose M_1 and M_2 are CTS's. Let

$$M_1 \times M_2 = ((G_1 \times G_2)^\dagger, \mathrm{id}_1 \times \mathrm{id}_2, \uparrow),$$

where $\uparrow$ is defined by the following condition:

- Suppose (f_1, f_2) and (g_1, g_2) are coinitial transitions of $(G_1 \times G_2)^\dagger$. If both pairs f_1, g_1 and f_2, g_2 are consistent, then $(f_1, f_2) \uparrow (g_1, g_2) = (f_1 \uparrow_1 g_1, f_2 \uparrow_2 g_2)$. If one of the pairs f_1, g_1 or f_2, g_2 is not consistent, then $(f_1, g_1) \uparrow (f_2, g_2) = \omega_{(\mathrm{cod}(f_2), \mathrm{cod}(g_2))}$.

We call $M_1 \times M_2$ the *product* of M_1 and M_2.

Proposition 3 *If M_1 and M_2 are CTS's, then $M_1 \times M_2$ is a CTS. Moreover, $M_1 \times M_2$, equipped with the evident projection morphisms, is a categorical product of M_1 and M_2 in* **CTS***.*

2.1.3 Equalizer

Suppose $M = (G^\dagger, \mathrm{id}, \uparrow)$ and N are CTS's, and $F_1, F_2 : M \to N$ are morphisms. Define

$$M[F_1 = F_2] = ((G')^\dagger, \mathrm{id}', \uparrow')$$

so that:

- The graph G' is the subgraph of G consisting of all proper states and proper transitions of M that have equal images under F_1 and F_2.

- The map id' is the restriction of id to $(G')^\dagger$.

- The operation $\uparrow'$ is the restriction to $(G')^\dagger$ of the operation $\uparrow$ on $G^\dagger$.

Note that for each q in G', the transition id_q has equal images under F_1 and F_2 by definition of a morphism, so that it is in fact in G'. Also, if f, g are coinitial transitions in $(G')^\dagger$, then the fact that $F_1(f) = F_2(f)$ and $F_1(g) = F_2(g)$, plus the fact that F_1 and F_2 are morphisms, shows that $F_1(f \uparrow g) = F_2(f \uparrow g)$, and hence $f \uparrow g$ is in $(G')^\dagger$.

Proposition 4 *If M is a CTS and $F_1, F_2 : M \to N$ are morphisms, then $M[F_1 = F_2]$ is a CTS. Moreover, the inclusion of $M[F_1 = F_2]$ in M is a categorical equalizer of F_1 and F_2 in* **CTS**.

2.2 Completion of a CTS

Every CTS M freely generates a complete CTS M^*, which has the same set of states as M and whose transitions can be thought of as the finite concurrent computations of M.

For this paper, we merely sketch the details of the construction. Given a CTS M, form G^*, the "transitive closure of," or the "free category generated by," the underlying graph G. The graph G has the same objects as M, and has as arrows the finite "composable" sequences of arrows of M (*i.e.* the ordinary finite computation sequences of M). Extend the translation operation $\uparrow$ by induction to G^*. The extended translation operation induces a congruence $\sim$ on G^*, where $f \sim g$ iff $f \uparrow g$ and $g \uparrow f$ are both sequences of identities. Factoring with respect to this congruence yields the CTS M^*. As a result of the construction, every morphism $F : M \to N$ extends uniquely to a morphism $F^* : M^* \to N$.

Proposition 5 *Every transition f of M^* is either an identity transition, or is of the form gh, where g is a transition of M^* and h is a transition of M.*

Define a property P of transitions of M^* to be *inductive* if whenever P holds for all proper prefixes of a transition f, then P holds for f as well. We have the following induction principle for M^*.

Proposition 6 *If P is an inductive property of transitions of M^*, then P holds for all transitions of M^*.*

It is helpful to notice that the arrows of a concurrent transition system have many of the same formal properties (except for the Church-Rosser property), if $f \uparrow g$ is interpreted as the "residual of the reduction f after the reduction g," as do the "one-step reductions" in the λ-calculus, and that the definition of $\sim$ generalizes the definition of "strongly equivalent" reduction sequences [Lev78,BL79,Bar81].

2.3 Computations as Ideals

A subset γ of a partially ordered set $(S, \preceq)$ is *directed* if γ is nonempty and every finite subset of γ has an upper bound in γ. A subset γ of S is *downward-closed* if for all $f \in \gamma$, if $g \preceq f$, then $g \in \gamma$. An *ideal* of $(S, \preceq)$ is a downward-closed, directed subset of S. It is easy to verify that if f is an element of S, then the set of all $g \in S$ such that $g \preceq f$ is an ideal. This ideal is called the *principal ideal* generated by f.

The completion M^* of a CTS M is interesting because it permits us to give a very convenient definition of the computations of M. Formally, suppose M is a CTS and q is a state of M. A *q-computation* of M is an ideal γ of $M^*(q, \cdot)$. A q-computation γ is *finite* if it is the principal ideal generated by a transition $f : q \to r$ of M^*. In this case, the state r is called the *final state* of γ. Computations that are not finite are called *infinite*. A standard consequence of the ideal construction is that the set of all q-computations of a CTS M, under inclusion order, is an algebraic directed-complete partial order, whose isolated elements are exactly the finite computations [Gue81].

Define a subset γ of $M^*(q, \cdot)$ to be *consistent* if for every finite subset δ of γ, the transition $\bigvee \delta$ is not ω_q.

Proposition 7 *Suppose γ is a consistent subset of $M^*(q, \cdot)$. Then there exists a least computation $\overline{\gamma}$ of M such that $\gamma \subseteq \overline{\gamma}$.*

3 Semantics of Concurrent Process Networks

In this section we define three semantic models for concurrent process networks. The "Kahn function" model is Kahn's denotational model for networks of determinate processes. The "input/output automaton" model, in which processes are modeled using a kind of generalized state-transition function, or "dynamics," has a classical automata-theoretic flavor. The "labeled process" model is an operational model similar to the "labeled transition system" models that have been used to describe the languages CSP and CCS.

3.1 History Monoids

We shall ultimately wish to assign, to each computation of a process, a corresponding "history" of the interesting occurrences in that computation. The input/output behavior of a process will then be obtained in terms of these histories. For example, in a dataflow model such as Kahn's, the interesting occurrences are the exchange of data values between processes. Alternatively, in CSP, the interesting occurrences are the communication events that are shared between processes. Usually, when ordinary transition systems are used as the underlying operational model, histories are defined to be sequences of events. For the concurrent transition system model, though, it is natural to use histories that are elements of a "history monoid," which we now define.

A *history monoid* is a monoid N with the following properties:

1. For all $u, v, w \in N$, if $uv = uw$, then $v = w$.

2. If $\sqsubseteq_N$ is the prefix relation induced by the monoid operation (*i.e.* $u \sqsubseteq_N v$ iff $\exists w(uw = v)$), then $\sqsubseteq_N$ is a partial order with respect to which each pair u, v with an upper bound has a least upper bound.

We use $\perp_N$ to denote the identity, $\sqsubseteq_N$ to denote the prefix relation, and $\sqcup_N$ to denote the least upper bound operation, of a history monoid N. Subscripts will be omitted when the monoid is clear from the context.

The *ideal space* of a history monoid N is the cpo $\overline{N}$ of ideals of N, ordered by inclusion.

An example of a history monoid is the free monoid Σ^* generated by an set Σ. The corresponding ideal space is the cpo Σ^∞ of all finite and infinite strings over Σ, with the prefix order. Thus, history monoids generalize the usual definition of histories as sequences of events. Another example of a history monoid is the monoid $[C \to \Sigma^*]$ of "finite channel histories," which are functions from a set C (of "channels") to Σ^*. The monoid identity and multiplication are inherited pointwise from Σ^*. The corresponding ideal space is the cpo $[C \to \Sigma^\infty]$.

It is easy to see that the set of transitions of a complete CTS with exactly one proper state, is a history monoid. Conversely, it can be shown that any history monoid can be made into the set of transitions of a one-proper-state complete CTS by taking the monoid identity as the proper identity transition id, letting u, v be consistent iff $u \sqcup v$ exists, and then defining $u \uparrow v = (u \sqcup v) \setminus v$, where $(u \sqcup v) \setminus v$ denotes the unique w such that $vw = (u \sqcup v)$. We may therefore consider the collection of history monoids as a full subcategory of CTS. It will be convenient to switch freely between the view of a history monoid as a monoid and as a CTS. When viewing a history monoid N as a CTS, we use $\top_N$ to denote the arrow from the single proper state of N to Ω.

History monoids are closely related to the "positive semirings" defined by Main and Benson [MB84]. Essentially, a history monoid N is a positive semiring in which there is a further connection between $+$ and $\cdot$; namely, $+$ is least upper bound with respect to the prefix order induced by $\cdot$, and in which a left-cancellation law holds for $\cdot$. History monoids are also closely related to the "synchronization algebras" of Winskel [Win84b,Win86], In fact, the set of proper transitions of a complete CTS with just one proper state q can be regarded as a synchronzation algebra if we identify Winskel's $*$ with our arrow id_q, and Winskel's operation $\bullet$ with our operation $\vee$. (The converse is not possible, in general, since complete CTS's are more highly structured than synchronization algebras.) We use history monoids below in the definition of connection of labeled processes in essentially the same way as Winskel uses synchronization algebras to define parallel composition of synchronization trees. We find it an advantage that the need for a separate notion of synchronization algebra is avoided.

3.2 Kahn Functions

Suppose X, Y are history monoids. An (X, Y)-*Kahn function* is a function $\phi : \overline{X} \to \overline{Y}$, which is continuous with respect to the cpo structure on $\overline{X}$ and $\overline{Y}$.

In this paper, we are concerned with three kinds of operations on processes:

- The *product* operation, with which processes are juxtaposed into a single network with each process running concurrently with and independently of the others.
- *Abstraction* operations, with which detail in a process or network is suppressed. (Example: hiding an internal channel.)

- *Connection* operations, with which processes are synchronized in various ways. (Example: "feeding back" an output channel to an input channel.)

A fourth class of interesting operations are the *recursion* operations, in which processes are built that (conceptually) have copies of themselves as components. We leave the treatment of recursion, and the related problem of networks whose structure changes dynamically during execution, to a future paper.

3.2.1 Product

Let ϕ_1 be an (X_1, Y_1)-Kahn function, and ϕ_2 an (X_2, Y_2)-Kahn function. The *product* of ϕ_1 and ϕ_2 is the $(X_1 \times X_2, Y_1 \times Y_2)$-Kahn function $\phi_1 \times \phi_2$.

3.2.2 Abstraction

Suppose ϕ is an (X, Y)-Kahn function, and $\rho : Y \to Y'$ is a morphism. The *abstraction* of ϕ by ρ is the Kahn function $\overline{\rho} \circ \phi$, where $\overline{\rho} : \overline{Y} \to \overline{Y}'$ is the unique continuous extension of ρ.

3.2.3 Connection

Suppose ϕ is an $(X \times C, Y)$-Kahn function, and $\rho : Y \to C$ is a morphism. The *connection* of ϕ by ρ is the (X, Y)-Kahn function $\phi[\rho]$ which is the least fixed point $\mu\Phi$ of the continuous functional

$$\Phi : [\overline{X} \to \overline{Y}] \to [\overline{X} \to \overline{Y}],$$

defined by $\Phi(\psi) = \phi \circ (\mathrm{id}_{\overline{X}} \times (\overline{\rho} \circ \psi))$.

3.3 Input/Output Automata

Suppose M is a CTS. An endomorphism $F : M \to M$ is *orthogonal* if for all transitions f of M, if $F(f)$ is an identity, then so is f. A *dynamics* is a monoid homomorphism $D : X \to \mathbf{CTS}_{\mathrm{o}}(M, M)$, where X is a history monoid and $\mathbf{CTS}_{\mathrm{o}}(M, M)$ is the monoid of orthogonal endomorphisms of M.

Suppose X and Y are history monoids. An (X, Y)-*input/output automaton* is a four-tuple $A = (M, I, D, O)$, where M is a CTS, called the *underlying CTS*, $D : X \to \mathbf{CTS}_{\mathrm{o}}(M, M)$ is a dynamics, $I : \mathbf{1} \to M$ is a morphism, called the *initial state map*, and $O : M \to Y$ is a morphism, called the *output map*, such that for each $x \in X$ we have $O \circ D(x) = O$. The elements of X are called the *input histories*, and the elements of Y are called the *output histories*, of A. An input/output automaton is called a *Kahn automaton* if its underlying CTS is determinate.

It is evident that the definition of a dynamics generalizes the classical automata-theoretic definition (*e.g.* [Eil74]) of a "state-transition function" or "action" as a monoid homomorphism $\delta : X \to \mathbf{Set}(Q, Q)$, or equivalently, as a map $\delta : X \times Q \to Q$ such that $\delta(\perp_X, q) = q$ and $\delta(xy, q) = \delta(y, \delta(x, q))$. [1] Our generalized dynamics acts not only on states of M, but also on transitions of M.

To exemplify the above definitions, we give a construction that yields, for each morphism $\phi : X \to Y$, an (X, Y)-Kahn automaton $A_\phi = (M, I, D, O)$.

Define M as follows:

- Proper states: all elements (x, y) of $X \times Y$.
- Proper transitions: all triples $(x, y, v) \in X \times Y \times Y$ such that either $v = \perp_Y$, or else $yv \sqsubseteq \phi(x)$. Define $\mathrm{dom}(x, y, v) = (x, y)$ and $\mathrm{cod}(x, y, v) = (x, yv)$.
- Proper identity transitions: all transitions

$$(x, y, \perp_Y) : (x, y) \to (x, y).$$

- Translation: (x, y, v) and (x, y, v') are consistent iff v and v' are consistent, in which case

$$(x, y, v) \uparrow (x, y, v') = (x, yv', v \uparrow v').$$

Let $I : \mathbf{1} \to M$ map the single state $\imath$ of $\mathbf{1}$ to the state $(\perp_X, \perp_Y)$ of M. Let $O : M \to Y$ map each transition (x, y, v) of M to $v \in Y$. Let $D : X \to \mathbf{CTS}(M, M)$ be defined so that if $u \in X$ and (x, y, v) is a transition of M, then $D(u)(x, y, v) = (xu, y, v)$.

We will show (Theorem 3), that the automaton A_ϕ actually has ϕ (more precisely, its continuous extension $\overline{\phi} : \overline{X} \to \overline{Y}$) as its input/output behavior.

Versions of the product, abstraction, and connection operations are easily defined for input/output automata.

---

[1] We would like to define a dynamics as a morphism $D' : X \times M \to M$, so that the definition becomes a special case of that of Arbib and Manes [AM75]. Unfortunately, we have in mind monoid homomorphisms $D : X \to \mathbf{CTS}(M, M)$, that do not correspond to morphisms $D' : X \times M \to M$. However, it is possible that replacing "$X \times \cdot$" by a slightly different endofunctor of $\mathbf{CTS}$ would rectify this problem.

3.3.1 Product

Suppose $A_1 = (M_1, I_1, D_1, O_1)$ is an (X_1, Y_1)-input/ output automaton and $A_2 = (M_2, I_2, D_2, O_2)$ is an (X_2, Y_2)-input/output automaton. The *product* of A_1 and A_2 is defined to be $A_1 \times A_2 = (M_1 \times M_2, I_1 \times I_2, D_1 \times D_2, O_1 \times O_2)$, which is easily seen to be an $(X_1 \times X_2, Y_1 \times Y_2)$-input/output automaton.

3.3.2 Abstraction

Suppose $A = (M, I, D, O)$ is an (X, Y)-input/output automaton and $\rho : Y \to Y'$ is a morphism. The *abstraction* of A by ρ is the (X, Y')-input/output automaton $\rho \circ A = (M, I, D, \rho \circ O)$.

3.3.3 Connection

Suppose $A = (M, I, D, O)$ is an $(X \times C, Y)$-input/output automaton, and $\rho : Y \to C$ is a morphism. The *connection* of A by ρ is the (X, Y)-input/output automaton $A[\rho] = (M', I', D', O')$, defined as follows:

- M' has as states the states of M, and as transitions the transitions of M, however domain and codomain are defined in M' so that $\mathrm{dom}'(f) = \mathrm{dom}(f)$ and $\mathrm{cod}'(f) = D(\perp_X, \rho \circ O(f))(\mathrm{cod}(f))$. Each identity transition id_q of M is also the identity transition id'_q of M'. Translation for M' is defined by the formula $f \uparrow' g = D(\perp_X, \rho \circ O(g))(f \uparrow g)$.
- $I' = I$, and $O' = O$.
- $D'(x)(f) = D(x, \rho \circ O(f))(f)$ for all $x \in X$ and all transitions f of M'.

Intuitively, the automaton $A[\rho]$ represents the automaton A with its output "fed back" through ρ to the C component of its input. Thus, each transition f' of $A[\rho]$ is a transition f of A that has been "composed" with the effect (given by $D(\perp_X, \rho \circ O(f))$) of the feedback input $\rho \circ O(f)$ associated with f.

Lemma 1 *If A is a $(X \times C, Y)$-Kahn automaton, and $\rho : Y \to C$ is a morphism, then $A[\rho]$ is an (X, Y)-Kahn automaton.*

Proof – Straightforward. ∎

3.4 Labeled Processes

A "labeled process" is formed by taking a CTS, designating an initial state, and assigning a label to each of its transitions. Labeled processes directly generalize "labeled transition systems," as used, for example, in [BR83].

Formally, suppose N is a history monoid. An *N-labeled process* is a triple $P = (M, I, L)$, where M is a CTS, called the *underlying CTS*, $I : \mathbf{1} \to M$ is a morphism, called the *initial state map*, and $L : M \to N$ is a morphism, called the *labeling map*.

3.4.1 Product

Suppose $P_1 = (M_1, I_1, L_1)$ is an N_1-labeled process, and $P_2 = (M_2, I_2, L_2)$ is an N_2-labeled process. The *product* of P_1 and P_2 is the $(N_1 \times N_2)$-labeled process $P_1 \times P_2 = (M_1 \times M_2, I_1 \times I_2, L_1 \times L_2)$.

3.4.2 Abstraction

Suppose $P = (M, I, L)$ is an N-labeled process, and $\rho : N \to N'$ is a morphism. The *abstraction of P by ρ* is the N'-labeled process $\rho \circ P = (M, I, \rho \circ L)$.

3.4.3 Connection

Suppose $P = (M, I, L)$ is an N-labeled process, and $\rho_1 : N \to C$ and $\rho_2 : N \to C$ are morphisms. Let $\epsilon : M' \to M$ be the equalizer of $\rho_1 \circ L$ and $\rho_2 \circ L$. The *connection of P by $\rho_1 = \rho_2$* is the N-labeled process $P[\rho_1 = \rho_2] = (M', I', L')$, where $L' = L \circ \epsilon$, and $I' : \mathbf{1} \to M'$ is the unique map such that $\epsilon \circ I' = I$, which is determined by the universal property of ϵ and the fact that $\rho_1 \circ L \circ I = \rho_2 \circ L \circ I$.

$$\begin{tikzcd}
\mathbf{1} \arrow[r, "I"] \arrow[dr, "I'"'] & M \arrow[r, "L"] & N \arrow[r, shift left, "\rho_1"] \arrow[r, shift right, "\rho_2"'] & C \\
& M' \arrow[u, "\epsilon"'] \arrow[ur, "L'"'] & &
\end{tikzcd}$$

4 Input/Output Automata Determine Labeled Processes

In this section we construct a mapping $\mathcal{L}$ that takes each (X,Y)-input/output automaton A to a corresponding $(X \times Y)$-labeled process $\mathcal{L}(A)$, and we show that this mapping preserves the process operations up to isomorphism of labeled processes. We also obtain a direct characterization of those $(X \times Y)$-labeled processes that are isomorphic to the image of an (X,Y)-input/output automaton, and those isomorphic to the image of an (X,Y)-Kahn automaton. We call the latter "Kahn processes."

Formally, suppose $A = (M, I, D, O)$ is an (X,Y)-input/output automaton.

Construct a CTS M' as follows:

- Proper States: the proper states of M.
- Proper Transitions: all pairs (x, f), where $x \in X$ and f is a proper transition of M. Define $\mathrm{dom}'(x, f) = \mathrm{dom}(f)$ and $\mathrm{cod}'(x, f) = \mathrm{cod}(D(x)(f))$.
- Identity Transitions: all pairs $(\perp_X, \mathrm{id}_q)$.
- Translation: (x, f) and (x', f') are consistent iff x, x' and f, f' are consistent, in which case $(x, f) \uparrow (x', f')$ is defined to be the pair $(x \uparrow x', D(x')(f \uparrow f'))$.

It is straightforward to use the defining properties of D to verify that M' indeed satisfies the axioms for a CTS. The assumption that $D(x)$ is orthogonal for each $x \in X$ is used to satisfy axiom (4) in the definition of a CTS.

Let $I' : \mathbf{1} \to M'$ map the unique state $\imath$ of $\mathbf{1}$ to the state $I(\imath)$ of M'. Let $L' : M' \to X \times Y$ map each transition (x, f) of M' to the element $(x, O(f))$ of $X \times Y$.

Then $\mathcal{L}(A) = (M', I', L')$ is an $(X \times Y)$-labeled process, which we call the labeled process *determined by* A.

Suppose $P = (M, I, L)$ and $P' = (M', I', L')$ are N-labeled processes. An *isomorphism* from P to P' is an isomorphism $F : M \to M'$ of the underlying CTS's, such that $I' = F \circ I$ and $O = O' \circ F$. We say that P and P' are *isomorphic*, and write $P \simeq P'$, if there exists an isomorphism from P to P'. If $P \simeq \mathcal{L}(A)$ for some (X,Y)-input/output automaton A, then we call P an (X,Y)-*input/output process*. If $P \simeq \mathcal{L}(A)$ for some (X,Y)-Kahn automaton A, then we call P an (X,Y)-*Kahn process*.

Suppose $P = (M, I, L)$ is an $(X \times Y)$-labeled process. We write L^X (resp. L^Y) for the X-component (resp. Y-component) of L. If P is an (X, Y)-input/output process, then we say that a transition f of M is a *pure output* transition if $L^X(f) = \perp_X$.

Theorem 1 *An $(X \times Y)$-labeled process $P = (M, I, L)$ is an (X, Y)-input/output process iff P has the following properties:*

1. *For each proper state q of M, and each $x \in X$, there exists a transition x_q of M, such that $\mathrm{dom}(x_q) = q$ and $L(x_q) = (x, \perp_Y)$, and such that if f is any transition of M with $\mathrm{dom}(f) = q$ and $x \sqsubseteq L^X(f)$, then $x_q \preceq f$. (We call the transitions x_q* pure input *transitions.)*
2. *For each proper transition $f : q \to r$ of M and each $x \in X$, we have $x_q \uparrow f = (x \uparrow L^X(f))_r$.*
3. *For each proper transition $f : q \to r$ of M, and each $x \in X$, if x and $L^X(f)$ are consistent, then x_q and f are consistent, and $x_q \vee f$ exists in M.*
4. *Every proper transition $f : q \to r$ of M, with $L^X(f) = x$, can be written as the join of a pure input and a pure output transition,* viz. *$f = x_q \vee f'$.*
5. *For each proper, pure output transition f of M and each $x \in X$, if $f \uparrow x_q$ is an identity transition, then f is an identity transition.*

Moreover, P is an (X, Y)-Kahn process iff P has the following property in addition to properties (1)-(5):

6. *For each coinitial pair f_1, f_2 of transitions of M, if $L^X(f_1)$ and $L^X(f_2)$ are consistent, then f_1 and f_2 are consistent.*

Proof – $\Rightarrow$ Suppose $P = (M_P, I_P, L_P) \simeq \mathcal{L}(A)$ for some (X, Y)-input/output automaton $A = (M, I, D, O)$. Since the properties (1)-(6) are preserved under isomorphism, we may suppose without loss of generality that $P = \mathcal{L}(A)$.

Property (1) is satisfied by taking $x_q = (x, \mathrm{id}_q)$ for each $x \in X$ and each state q of M_P. Property (2) is then obvious from the definition of $\uparrow$ on M_P.

To show property (3), recall that each transition $f : q \to r$ of M_P is a pair $(L_P^X(f), f')$, where f' is an arrow of M with $\mathrm{dom}(f') = q$. If x and $L_P^X(f)$ are consistent, then

$(L_P^X(f), f')$ and $x_q = (x, \mathrm{id}_q)$ are consistent by definition of translation for M_P, and $x_q \vee f = (x \sqcup L_P^X(f), f')$.

Property (4) holds because each transition $f : q \to r$ of M_P, with $L_P^X(f) = x$, is a pair (x, f'), hence can be written $f = (x, \mathrm{id}_q) \vee (\perp_X, f')$, that is, $f = x_q \vee (\perp_X, f')$.

To show property (5), suppose f is a proper, pure output transition of M_P, and $f \uparrow x_q$ is an identity transition. Then $x_q = (x, \mathrm{id}_q)$ and $f = (\perp_X, f')$ for some f', so $f \uparrow x_q = (\perp_X, D(x)(f'))$. If $(\perp_X, D(x)(f'))$ is an identity transition, then $D(x)(f')$ is an identity transition, which implies f' is an identity transition by the orthogonality of $D(x)$. Hence f is an identity transition.

Finally, to show (6), if P is an (X, Y)-Kahn process, then the CTS M is determinate, which means that any two coinitial transitions of M are consistent. Thus, if f_1, f_2 are transitions of M, with $L^X(f_1)$ and $L^X(f_2)$ consistent, then $f_1 = (x_1, f_1')$ and $f_2 = (x_2, f_2')$, where x_1, x_2 are consistent, and f_1', f_2' are coinitial, hence consistent. It follows by definition of translation for M_P that f_1 and f_2 are consistent.

$\Leftarrow$ Suppose P has properties (1)-(5). We show how to construct $A = (M, I, D, O)$ so that $P \simeq \mathcal{L}(A)$, and so that A is determinate if P has property (6).

It is straightforward to see that the states of M_P, equipped with all pure output transitions of M_P, form a sub-CTS M of M_P. Let $I : \mathbf{1} \to M$ be defined by $I(\imath) = I_P(\imath)$. Let $O : M \to Y$ be the restriction of L_P^Y to M. To define D, note that if f is a pure output transition of M_P with $\mathrm{dom}(f) = q$, then for each $x \in X$ the transition $f \uparrow x_q$ is also a pure output transition, because O is a morphism. Thus, for each $x \in X$, and each transition f of M with $\mathrm{dom}(f) = q$, the transition $f \uparrow x_q$ is a transition of M. Define D by $D(x)(f) = f \uparrow x_{\mathrm{dom}(f)}$.

Note that for all $x \in X$ and all pure output transitions f, we have $O(D(x)(f)) = O(f \uparrow x_{\mathrm{dom}(f)}) = O(f) \uparrow x = O(f)$, hence $O \circ D(x) = O$. We claim further that D is a dynamics, hence $A = (M, I, D, O)$ is an input/output automaton. The orthogonality of $D(x)$ for all $x \in X$ is immediate from the definition of D and property (5).

We first show that for each $x \in X$, the map $D(x) : M \to M$ is a morphism. Since by property (1), $L_P(x_q) = (x, \perp_Y) \neq \top_{X \times Y}$, and thus $x_q \neq \omega_q$, we know that $D(x)(q) = \Omega$ implies $q = \Omega$. Moreover, $D(x)(\mathrm{id}_q) = \mathrm{id}_q \uparrow x_q = \mathrm{id}_{D(x)(q)}$ and $D(x)(\omega_q) = \omega_q \uparrow x_q = \omega_{D(x)(q)}$. If f, f' are coinitial pure output transitions with $f' : q \to r$, then $D(x)(f \uparrow f') = (f \uparrow f') \uparrow x_r = (f \uparrow f') \uparrow (x_q \uparrow f')$, because $x_q \uparrow f' = x_r$ by property (2). Also, $(f \uparrow f') \uparrow (x_q \uparrow f') =$

$(f \uparrow x_q) \uparrow (f' \uparrow x_q) = D(x)(f) \uparrow D(x)(f')$.

We next show that $D : X \to \mathbf{CTS}(M, M)$ is a monoid homomorphism. That $D(\perp_X) = \mathrm{id}_M$ is clear from the definition of D and because property (1) implies that $(\perp_X)_q = \mathrm{id}_q$ holds for all states q. That $D(xx')(f) = D(x')(D(x)(f))$ follows from the fact that $(xx')_q = x_q x'_{\mathrm{cod}(x_q)}$. This fact, in turn, holds because $x_q \preceq (xx')_q$ by property (1) and $(xx')_q \uparrow x_q = x'_{\mathrm{cod}(x_q)}$ by property (2).

We next claim that $P \simeq \mathcal{L}(A)$. Let $(M', I', L') = \mathcal{L}(A)$. Let $F : M' \to M_P$ be the morphism that takes each state of M' to the same state of M_P, and each transition $(x, f) : q \to r$ of M' to the transition $x_q \vee f : q \to r$ of M_P, which exists by property (3). This map is clearly a bijection on states. It is surjective on transitions because by property (4), every transition $f : q \to r$ of M_P, with $L^X_P(f) = x$, can be written $f = x_q \vee f'$, where f' is a pure output transition. The morphism F is injective on transitions because for coinitial pure output transitions f, f' of M_P, with domain q, if $f \vee x_q = f' \vee x_q$, then $(f \vee x_q) \uparrow (f' \vee x_q)$ is an identity transition. Since $(f \vee x_q) \uparrow (f' \vee x_q) = [f \uparrow (f' \vee x_q)] \vee [x_q \uparrow (f' \vee x_q)]$, it follows that $f \uparrow (f' \vee x_q)$ is an identity transition. However, $f \uparrow (f' \vee x_q) = (f \uparrow f') \uparrow (x_q \uparrow f')$, and $x_q \uparrow f' = x_{\mathrm{cod}(f')}$ by property (2). Hence $f \uparrow f'$ is an identity transition by property (5). Similar reasoning shows that $f' \uparrow f$ is an identity transition, thus $f = f'$.

Finally, we claim that A is determinate if P has property (6). But this is clear, since given coinitial transitions f, f' of A, we know that f, f' are pure output transitions of M_P, hence are consistent if P has property (6). ∎

Theorem 2 *The map $\mathcal{L}$ has the following properties:*

1. *Suppose A_1 is an (X_1, Y_1)-input/output automaton, and A_2 is an (X_2, Y_2)-input/output automaton. Then $\mathcal{L}(A_1 \times A_2) \simeq \mathcal{L}(A_1) \times \mathcal{L}(A_2)$.*
2. *Suppose A is an (X, Y)-input/output automaton, and $\rho : Y \to Y'$ is a morphism. Then $\mathcal{L}(\rho \circ A) \simeq \rho \circ \mathcal{L}(A)$.*
3. *Suppose A is an $(X \times C, Y)$-input/output automaton, and $\rho : Y \to C$ is a morphism. Then $\mathcal{L}(A[\rho]) \simeq \mathcal{L}(A)[\pi_C = \rho]$, where $\pi_C : X \times C \to C$ is the projection morphism associated with the product $X \times C$.*

Proof – We prove only (3), the proofs of (1) and (2) are straightforward.

(3) Let $A = (M, I, D, O)$ and $A[\rho] = (M', I', D', O')$. Let $P = \mathcal{L}(A) = (M_P, I_P, L_P)$, and let $P' = \mathcal{L}(A[\rho]) =$

$(M_{P'}, I_{P'}, L_{P'})$. The CTS's M, M', M_P, and $M_{P'}$ all have exactly the same states. The transitions from q to r of $M_{P'}$ are all pairs (x, f), with $x \in X$ and f a transition of M', such that $q = \mathrm{dom}'(f)$ and $r = \mathrm{cod}'(D'(x)(f))$, and the transitions from q to r of M_P are all pairs $((x, c), f)$, with $(x, c) \in X \times C$ and f a transition of M, such that $q = \mathrm{dom}(f)$ and $r = \mathrm{cod}(D(x, c)(f))$. But then it is clear, because $\mathrm{dom}'(f) = \mathrm{dom}(f)$, $\mathrm{cod}'(f) = D(\bot_X, \rho \circ O(f))(\mathrm{cod}(f))$, and $D'(x)(f) = D(x, \rho \circ O(f))(f)$, that the map F that takes each state of $M_{P'}$ to the same state of M_P and each transition (x, f) of $M_{P'}$ to the transition $((x, \rho \circ O(f)), f)$ of M_P, is an isomorphism of $M_{P'}$ to the sub-CTS of M_P which has the same states as M_P, and which has as transitions all tuples $((x, c), f)$ with $c = \rho \circ O(f)$. But this sub-CTS of M_P is exactly the underlying CTS of $\mathcal{L}(A)[\pi_C = \rho]$. It is easily verified that $I_P = F \circ I_{P'}$ and $O_{P'} = O_P \circ F$, as required for F to be an isomorphism of labeled processes. ∎

5 Kahn Processes Determine Kahn Functions

In this section, we construct a mapping $\mathcal{K}$ that maps each (X, Y)-Kahn process to a corresponding (X, Y)-Kahn function (its "input/output behavior"), and we show that the process operations are preserved by this mapping.

Lemma 2 *Suppose $P = (M, I, L)$ is an (X, Y)-input/output process. Then the following hold:*

1. *(Decomposability) Every proper transition $f : q \to r$ of M, with $L^X(f) = x$, can be decomposed $f = x_q \vee f'$, where f' is a pure output transition.*
2. *(Receptivity) If $f : q \to r$ is a transition of M^*, and $x \in X$ is such that x and $L^X_P(f)$ are consistent, then f and x_q are consistent.*
3. *(Factorability) Every transition $f : q \to r$ of M^*, with $L^X(f) = x$, can be factored into a pure input transition and a pure output transition,* viz.*: $f = x_q(f \uparrow x_q)$.*

Moreover, if P is an (X, Y)-Kahn process, then we have also:

4. *(Determinacy) If f, f' are coinitial transitions of M^* such that $L^X(f)$ and $L^X(f')$ are consistent, then f and f' are consistent.*

Proof – Decomposability is just property 4 of Theorem 1. Receptivity is proved by an induction on f (justified by Proposition 6), using properties (2) and (3) of Theorem 1. Determinacy is established by a (double) induction on (f, f'), using property (6) of Theorem 1.

Factorability is proved by induction on f. Suppose we have shown that all proper prefixes of $f : q \to r$ can be factored as claimed. If f is an identity, then we can write $f = (\perp_X)_q f$. If f is not an identity, then $f = gh$, where $g : q \to s$ is a transition of M^* and $h : s \to r$ is a transition of M. By induction, we can write $g = x'_q g'$, where $x' = L^X(g)$ and g' is a pure output transition. Suppose $x'_q : q \to p$ and $g' : p \to s$. By property (4) of Theorem 1, we can write $h = x''_s \vee h'$, where $x''_s = L^X(h)$. By property (2) of Theorem 1, x''_p and g' are consistent, and $x''_p \uparrow g' = x''_s$. Thus we have commutativity of the diagram below, showing that we can factor f as $f = gh = (x'_q x''_p)((g \uparrow x''_p)(h' \uparrow x''_s))$, as required. ∎

$$\begin{tikzcd}
& & t \arrow[dr, "g' \uparrow x''_p"] & & \\
& p \arrow[ur, "x''_p"] \arrow[dr, "g'"] & & u \arrow[dr, "h' \uparrow x''_s"] & \\
q \arrow[ur, "x'_q"] \arrow[rr, "g"] & & s \arrow[ur, "x''_s"] \arrow[rr, "h"] & & r
\end{tikzcd}$$

Suppose P is an N-labeled process. Each consistent set γ of transitions of M_P^* defines a consistent set $L_P(\gamma) \subseteq N$, which in turn extends to a least ideal $\overline{L}_P(\gamma) \in \overline{N}$. We call $\overline{L}_P(\gamma)$ the *history* of γ. If P is an (X, Y)-input/output process, and $\overline{L}_P(\gamma) = (x, y)$, then we call x the *input history*, and y the *output history*, of γ. The set γ is *maximal for input history* x if x is the input history of γ, and $\gamma' \subseteq \gamma$ whenever γ' is any consistent set of transitions of M_P^* with input history x. It is easy to see that consistent sets that are maximal for an input history, must be computations.

Lemma 3 *Suppose $P = (M, I, L)$ is an (X, Y)-input/output process. Then for each $x \in \overline{X}$, and each state q of M, there is a q-computation $\gamma_q(x)$ of M that is maximal for input x. Moreover, if P is a Kahn process, then $\gamma_q(x)$ is uniquely determined by x and q.*

Proof – Apply Zorn's Lemma to $\{x'_q : x' \sqsubseteq x, x' \in X\}$. ∎

From Lemma 3 it follows that each (X, Y)-Kahn process P determines a function $\mathcal{K}(P) : \overline{X} \to \overline{Y}$ (its input/output map) that takes each $x \in \overline{X}$ to $\overline{L}_P^Y(\gamma_{I(\imath)}(x))$.

Lemma 4 *If P is an (X, Y)-Kahn process, then $\mathcal{K}(P)$ is an (X, Y)-Kahn function.*

Proof –

Suppose $P = (M, I, L)$ is an (X, Y)-Kahn process. To see that $\mathcal{K}(P)$ is monotonic, note that if $x, x' \in \overline{X}$ are such that $x \sqsubseteq x'$, then $\gamma_{I(\imath)}(x) \subseteq \gamma_{I(\imath)}(x')$, and hence $\mathcal{K}(P)(x) \sqsubseteq \mathcal{K}(P)(x')$.

To see that $\mathcal{K}(P)$ is continuous, suppose we are given a directed collection $\{x_\alpha : \alpha \in A\}$ of elements of $\overline{X}$, with $\bigsqcup\{x_\alpha : \alpha \in A\} = x$. For each $\alpha \in A$, let γ_α be the unique $I(\imath)$-computation of P that is maximal for input history x_α, and let $y_\alpha = \mathcal{K}(P)(x_\alpha)$ be the output history of γ_α. The determinacy of P implies that the collection of computations $\Gamma = \{\gamma_\alpha : \alpha \in A\}$ is directed, hence its union is a computation γ whose input history is x, and whose output history is $y = \bigsqcup\{y_\alpha : \alpha \in A\}$. We claim that γ is in fact maximal for input history x, thus showing that $\mathcal{K}(P)(x) = y$. If γ were not maximal, then there would be a transition f of M^*, with $L_P^X(f) \sqsubseteq x$, but with $f \notin \gamma$. But since $L^X(f)$ is an isolated element of $\overline{X}$, it must be the case that $L^X(f) \sqsubseteq x_\alpha$ for some $\alpha \in A$. By the determinacy of P and the maximality of γ_α, we must have $f \in \gamma_\alpha$, contradicting the assumption that $f \notin \gamma$. ∎

The following lemma uses factorability to give a useful characterization of $\mathcal{K}(P)$. A corollary lets us determine $\mathcal{K}(\mathcal{L}(A))$ directly from A.

Lemma 5 *Suppose $P = (M, I, L)$ is an (X, Y)-Kahn process. Then for all $x \in \overline{X}$,*

$$\begin{aligned}\mathcal{K}(P)(x) \;=\; & \bigsqcup\{L^Y(f') : f' \text{ in } M^* \wedge L^X(f') = \bot_X \\ & \wedge\, \exists x' \in X(x' \sqsubseteq x \wedge \mathrm{dom}(f') = \mathrm{cod}(x'_{I(\imath)}))\}.\end{aligned}$$

Proof – Straightforward. ∎

Corollary 6 *Suppose $A = (M, I, D, O)$ is a determinate (X, Y)-input/output automaton. Then for all $x \in \overline{X}$,*

$$\begin{aligned}\mathcal{K}(\mathcal{L}(A))(x) &= \bigsqcup\{O(f') : f' \text{ in } M^* \\ &\qquad \wedge \exists x' \sqsubseteq x(\mathrm{dom}(f') = D(x')(I(\imath)))\}.\end{aligned}$$

Theorem 3 *Every (X, Y)-Kahn function ϕ is $\mathcal{K}(P)$ for some (X, Y)-Kahn process P.*

Proof – Suppose ϕ is an (X, Y)-Kahn function. Let $A_\phi = (M, I, D, O)$ be the determinate (X, Y)-input/output automaton constructed in Section 3.3. We claim that $\mathcal{K}(\mathcal{L}(A)) = \phi$. But this follows directly from Corollary 6, since

$$\begin{aligned}\{O(f') \ &: \ f' \text{ in } M^*, \exists x' \sqsubseteq x(\mathrm{dom}(f') = D(x')(I(\imath)))\} \\ &= \{y \in Y : y \sqsubseteq \phi(x)\}.\end{aligned}$$

∎

Theorem 4 *The map $\mathcal{K}$ has the following properties:*

1. *Suppose P_1 is an (X_1, Y_1)-Kahn process and P_2 is an (X_2, Y_2)-Kahn process. Then $P_1 \times P_2$ is an $(X_1 \times X_2, Y_1 \times Y_2)$-Kahn process, and $\mathcal{K}(P_1 \times P_2) = \mathcal{K}(P_1) \times \mathcal{K}(P_2)$.*
2. *Suppose P is an (X, Y)-Kahn process, and $\rho : Y \to Y'$ is a morphism. Then $\rho \circ P$ is an (X, Y')-Kahn process, and $\mathcal{K}(\rho \circ P) = \rho \circ \mathcal{K}(P)$.*
3. *(Kahn Principle) Suppose P is an $(X \times C, Y)$-Kahn process, and $\rho : Y \to C$ is a morphism. Then $P[\pi_C = \rho]$ is an (X, Y)-Kahn process, and $\mathcal{K}(P[\pi_C = \rho]) = \mathcal{K}(P)[\rho]$.*

We omit the straightforward proofs of (1) and (2). The proof of (3) uses two lemmas, which we prove first. In what follows, suppose P is an $(X \times C, Y)$-Kahn process, let $\rho : Y \to C$ be a morphism, and let $P' = P[\pi_C = \rho]$. Let $\phi = \mathcal{K}(P)$, and let $\Phi : (\overline{X} \to \overline{Y}) \to (\overline{X} \to \overline{Y})$ be defined by $\Phi(\psi) = \phi \circ (\mathrm{id}_{\overline{X}} \times (\overline{\rho} \circ \psi))$. Let ϕ^0 be the identically $\perp_{\overline{Y}}$ function, and for each $k \geq 0$, let $\phi^{k+1} = \Phi(\phi^k)$. Then $\mathcal{K}(P)[\rho] = \mu\Phi$, and the characterization $\mu\Phi = \bigsqcup_{k=0}^{\infty} \phi^k$ is standard.

Lemma 7 *For all proper transitions f of $M^*_{P'}$, with $\mathrm{dom}(f) = I_{P'}(\imath)$, there exists $k \geq 0$ such that $L^Y_{P'}(f) \sqsubseteq \phi^k(L^X_{P'}(f))$.*

Proof – The proof is by induction on f. Suppose we have established the result for all proper prefixes of f. If f is an identity transition, then the result clearly holds with $k = 0$.

If f is not an identity transition, then we can write $f = gh$ for some transition g of $M^*_{P'}$ and some transition h of $M_{P'}$. Then h is also a transition of M_P (because $M_{P'}$ is a sub-CTS of M_P), so by the decomposability of P, we can write $h = h^{\text{in}} \vee h^{\text{out}}$, where $L_P(gh^{\text{in}}) = (L_P^{X\times C}(gh), L_P^Y(g))$ and $L_P(gh^{\text{out}}) = (L_P^{X\times C}(g), L_P^Y(gh))$.

Applying the induction hypothesis to the proper prefix g of f, we obtain k such that $L^Y_{P'}(g) \sqsubseteq \phi^k(L^X_{P'}(g))$. Then $L^Y_{P'}(f) = L^Y_P(f) = L^Y_P(gh^{\text{out}}) \sqsubseteq \phi(L_P^{X\times C}(gh^{\text{out}}))$ because $\phi = \mathcal{K}(P)$. Moreover, $\phi(L_P^{X\times C}(gh^{\text{out}})) = \phi(L_P^{X\times C}(g)) = \phi(L^X_{P'}(g), \rho(L^Y_{P'}(g)))$ because $L^C_P(g) = \rho(L^Y_P(g)) = \rho(L^Y_{P'}(g))$ by definition of $M_{P'}$. Now, $\phi(L^X_{P'}(g), \rho(L^Y_{P'}(g))) \sqsubseteq \phi(L^X_{P'}(g), \rho \circ \phi^k(L^X_{P'}(g))) = \phi^{k+1}(L^X_{P'}(g))$, using the induction hypothesis, the monotonicity of ϕ and ρ, and the definition of ϕ^{k+1}. Finally, $\phi^{k+1}(L^X_{P'}(g)) \sqsubseteq \phi^{k+1}(L^X_{P'}(gh))$, by monotonicity of $L_{P'}$ and ϕ^{k+1}. ∎

Lemma 8 *For all $k \geq 0$, and all $(x, y) \in X \times Y$ with $y \sqsubseteq \phi^k(x)$, there exists a proper transition f of $M^*_{P'}$, with* $\text{dom}(f) = I_{P'}(\imath)$*, such that $x = L^X_{P'}(f)$ and $y \sqsubseteq L^Y_{P'}(f)$.*

Proof – The proof is by induction on k. If $k = 0$, then $\phi^k(x) = \bot_Y$, hence taking $f = x_{I(\imath)}$ suffices for the lemma.

Suppose now that the lemma has been established for some $k \geq 0$, and consider the case for $k + 1$. Suppose we are given $(x, y) \in X \times Y$ with $y \sqsubseteq \phi^{k+1}(x)$. By definition of ϕ^{k+1}, we have $y \sqsubseteq \phi(x, \overline{\rho} \circ \phi^k(x))$. By the continuity of $\overline{\rho}$ and ϕ, there exists $y' \in Y$ with $y' \sqsubseteq \phi^k(x)$, such that $y \sqsubseteq \phi(x, \rho(y'))$. Applying the induction hypothesis to (x, y') yields a transition f' of $M^*_{P'}$ with $x = L^X_{P'}(f')$ and $y' \sqsubseteq L^Y_{P'}(f')$.

Since f' is a transition of $M^*_{P'}$, we know that $L^C_P(f') = \rho(L^Y_P(f'))$, and hence $\rho(y') \sqsubseteq L^C_P(f')$. Since $\phi = \mathcal{K}(P)$, the output history of the unique $I_P(\imath)$-computation of M_P on input $L_P^{X\times C}(f')$ must have $\phi(x, \rho(y'))$, and hence y, as a prefix. Thus, there must exist a transition h of M^*_P, such that $L_P^{X\times C}(h) = \bot_{X\times C}$ and $y \sqsubseteq L^Y_P(f'h)$. But then using receptivity to form $h' = h \vee (\rho(L^Y_P(h)))_{\text{dom}(h)}$ yields a transition $f = f'h'$ of $M^*_{P'}$ with $L^X_{P'}(f) = L^X_{P'}(f') = x$ and $y \sqsubseteq L^Y_P(f'h') = L_{P'}(f)$, as required. ∎

Proof – (of the Kahn Principle, Theorem 4, part (3)) Suppose $P = \mathcal{L}(A)$, where A is an $(X \times C, Y)$-Kahn automaton. Then $A[\rho]$ is an (X, Y)-Kahn automaton by (Lemma

1). Since $P[\pi_C = \rho] \simeq \mathcal{L}(A[\rho])$ by Theorem 2 (3), it follows that $P[\pi_C = \rho]$ is an (X, Y)-Kahn process.

To establish the characterization of $\mathcal{K}(P[\pi_C = \rho])$, suppose $x \in \overline{X}$ is given. Then $\mathcal{K}(P[\pi_C = \rho])(x) = y$ iff y is the output history of the unique $I_{P'}(\imath)$-computation of P' which is maximal for input history x. This computation is precisely the set of all transitions f of $M^*_{P'}$ with $\mathrm{dom}(f) = I_{P'}(\imath)$ and $L^X_{P'}(f) \sqsubseteq x$.

By Lemma 7, to each transition f of $M^*_{P'}$, with $\mathrm{dom}(f) = I_{P'}(\imath)$, there is a $k \geq 0$ such that $L^Y_{P'}(f) \sqsubseteq \phi^k(L^X_{P'}(f))$. This shows that $L^Y_{P'}(f) \sqsubseteq \mu\Phi(x) = \mathcal{K}(P)[\rho](x)$, and hence $\mathcal{K}(P')(x) \sqsubseteq \mathcal{K}(P)[\rho](x)$.

To show that $\mathcal{K}(P)[\rho](x) \sqsubseteq \mathcal{K}(P')(x)$, note that by Lemma 8, for each $k \geq 0$ and $(x, y) \in X \times Y$, with $y \sqsubseteq \phi^k(x)$, there exists a transition f of $M^*_{P'}$, with $\mathrm{dom}(f) = I_{P'}(\imath)$, such that $x = L^X_{P'}(f)$ and $y \sqsubseteq L^Y_{P'}(f)$. In light of the characterization $\mu\Phi = \bigsqcup_{k=0}^{\infty} \phi^k$, it follows that the output history of the unique $I_{P'}(\imath)$-computation of $M_{P'}$ that is maximal for input x is at least $\mu\Phi(x)$. Hence $\mathcal{K}(P)[\rho](x) = \mu\Phi(x) \sqsubseteq \mathcal{K}(P')(x)$. ∎

6 Conclusion

The author was led to define concurrent transition systems because of his frustrated attempts to use ordinary transition systems to solve the basic problems considered in this paper: to obtain a simple transition system characterization of the "dataflow-like" processes with functional behavior, and to show directly from the transition system definition that such processes obey the Kahn Principle. Ordinary transition systems seemed not to express enough about concurrency to permit these results to go through, whereas with concurrent transition systems things are fairly smooth. In retrospect, when one considers the fundamental way in which the notions of join and translation are used in Theorem 1, it is hard to see how such a characterization could be expressed without them.

There are tantalizing hints of relationships between concurrent transition systems and well-established models of concurrency such as Petri Nets [Rei85], event structures [NPW81,Win84a,Win86], and other related models, such as the "behavior algebras" of [Win82,Win80]. As yet, we have only a limited understanding of the connections. A CTS can be obtained from a "net" [Thi86] by letting the states be the "cases" of the net and the transitions from q to

r be the "independent" sets u of events such that u is a step enabled in q and such that u leads from q to r. Coinitial transitions u and v are consistent iff $u \cup v$ is independent, in which case $u \uparrow v$ is the step $u \setminus v$. Given a CTS M with a designated initial state q, it is straightforward to make the set $M^*(q, \cdot)$ into an event structure (as defined in [Win86]) by defining the "consistent" sets of arrows to be the finite sets that are consistent as we have defined in this paper, and defining a consistent set $\gamma \subset M^*(q, \cdot)$ to "enable" an arrow f iff there is a subset δ of γ such that $(\bigvee \delta)h = f$ for some arrow h of M. Conversely, the set of configurations of an event structure is a partially ordered set in which every finite subset with an upper bound has a least upper bound, and hence is easily made into a complete CTS, by taking configurations as states and the ordering relationships as transitions. In a sense, concurrent transition systems can be thought of as a somewhat more primitive operational model than event structures, since in the former one is free to designate the set of states, whereas in the latter, states are always obtained as configurations.

The method of comparing semantic models we have used in this paper is similar to that used in [NPW81], but has some important differences with the more elaborate method proposed by Winskel [Win84a]. Winskel tries to find notions of "process morphism" such that a model becomes a category in which interesting process operations (such as parallel composition) become instances of categorical constructions (such as product). He compares models by finding adjunctions (such as coreflections) between categories. In contrast, we haven't necessarily tried to make processes into a category. Although there is an obvious definition of a morphism between labeled processes with the same labeling monoid, it is not clear that extensions to morphisms between processes with different labeling monoids are particularly useful. Instead, we have tried to construct a simple and intuitive operational semantics of concurrency (the labeled process model) entirely within the category of concurrent transition systems. We regard a model not as a category but as merely an algebra, and we attempt to compare models by constructing homomorphisms between them.

A number of extensions to the results of this paper naturally suggest themselves as topics for future work. Extensions to the labeled process model to include additional operations such as inverse image, nondeterministic union, and recursive definition, should be straightforward. It would also be interesting to try to characterize the power of the full input/output process model for representing nondeterminate dataflow networks. Although it is possible to represent "unfair merge" as an input/output automaton, "fair merge" apparently cannot be so represented. Thus, although the use of CTS's has obviated the need to consider "artificial fairness" arising from infinite computations coupled with an interleaved model of concurrency, it has not eliminated the need for information about "true fairness," such as that displayed by fair merge. We are currently investigating natural ways to incorporate such information into the input/output process model.

References

[AM75] M. A. Arbib and E. G. Manes. *Arrows, Structures, and Functors: The Categorical Imperative.* Academic Press, 1975.

[BA81] J. D. Brock and W. B. Ackerman. Scenarios: a model of non-determinate computation. In *Formalization of Programming Concepts*, pages 252–259, Springer-Verlag, 1981.

[Bar81] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* Volume 103 of *Studies in Logic and the Foundations of Mathematics*, North-Holland, 1981.

[BL79] G. Berry and J.-J. Lévy. Minimal and optimal computations of recursive programs. *Journal of the ACM*, 26(1):148–175, January 1979.

[BM82] R. J. Back and N. Mannila. A refinement of Kahn's semantics to handle nondeterminism and communication. In *Proc. ACM Symposium on Distributed Computing*, pages 111–120, 1982.

[BR83] S. D. Brookes and W. C. Rounds. Behavioral equivalence relations induced by programming logics. In *Proceedings of ICALP 83*, 1983.

[Bro83] M. Broy. Fixed point theory for communication and concurrency. In D. Bjørner, editor, *Formal Description of Programming Concepts II*, pages 125–148, North-Holland, 1983.

[Cad72] J. M. Cadiou. *Recursive Definitions of Partial Functions and Their Computations.* PhD thesis, Stanford University, 1972.

[Eil74] S. Eilenberg. *Automata, Languages, and Machines.* Volume A, Academic Press, 1974.

[Fau82] A. A. Faustini. An operational semantics for pure dataflow. In *Automata, Languages, and Programming, 9th Colloquium*, pages 212–224, Springer-Verlag, 1982.

[Gue81] I. Guessarian. *Algebraic Semantics.* Volume 99 of *Lecture Notes in Computer Science*, Springer Verlag, 1981.

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–676, 1978.

[Kah74] G. Kahn. The semantics of a simple language for parallel programming. In J. L. Rosenfeld, editor, *Information Processing 74*, North-Holland, 1974.

[Kel78] R. M. Keller. Denotational models for parallel programs with indeterminate operators. In E. J. Neuhold, editor, *Formal Description of Programming Concepts*, pages 337–366, North-Holland, 1978.

[KM77] G. Kahn and D. B. MacQueen. Coroutines and networks of parallel processes. In B. Gilchrist, editor, *Information Processing 77*, North-Holland, 1977.

[Kok86] J. N. Kok. Denotational semantics of nets with nondeterminism. In *ESOP 86*, pages 237–249, Springer-Verlag, March 1986.

[KP84] R. M. Keller and P. Panangaden. Semantics of networks containing indeterminate operators. In *Seminar on Concurrency*, pages 479–496, Springer-Verlag, 1984.

[Lev78] J.-J. Lévy. *Réductions Correctes et Optimales dans le Lambda Calcul.* PhD thesis, Université Paris VII, 1978.

[Mac79] D. B. MacQueen. *Models for Distributed Computing.* Technical Report 351, INRIA, 1979.

[MB84] M. G. Main and D. B. Benson. Functional behavior of nondeterministic and concurrent programs. *Information and Control*, 62:144–189, 1984.

[Mil80] R. Milner. *A Calculus of Communicating Systems.* Volume 92 of *Lecture Notes in Computer Science*, Springer Verlag, 1980.

[ML71] S. Mac Lane. *Categories for the Working Mathematician.* Volume 5 of *Graduate Texts in Mathematics*, Springer Verlag, 1971.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures, and domains, part I. *Theoretical Computer Science*, 13:85–108, 1981.

[Par82] D. M. R. Park. The "fairness problem" and nondeterministic computing networks. In *Proceedings, 4th Advanced Course on Theoretical Computer Science*, Mathematisch Centrum, 1982.

[Pra82] V. R. Pratt. On the composition of processes. In *Ninth Annual ACM Symposium on Principles of Programming Languages*, 1982.

[Rei85] W. Reisig. *Petri Nets.* Volume 4 of *EATCS Monographs on Theoretical Computer Science*, Springer, 1985.

[SAM83] M. Steenstrup, M. A. Arbib, and E. G. Manes. Port automata and the algebra of concurrent processes. *JCSS*, 27(1):29–50, 1983.

[SN85] J. Staples and V. L. Nguyen. A fixpoint semantics for nondeterministic data flow. *Journal of the ACM*, 32(2):411–444, April 1985.

[Sta86] E. W. Stark. *The Computation Category of a Concurrent Transition System.* Technical Report 86/08, State University of New York at Stony Brook Computer Science Dept., May 1986. Submitted to *Theoretical Computer Science*.

[Thi86] P. S. Thiagarajan. Elementary net systems. In *Advanced Course on Petri Nets*, GMD, Bad Honnef, September 1986.

[Win80] J. Winkowski. Behaviors of concurrent systems. *Theoretical Computer Science*, 12:39–60, 1980.

[Win82] J. Winkowski. An algebraic description of system behaviors. *Theoretical Computer Science*, 21:315–340, 1982.

[Win84a] G. Winskel. Categories of models for concurrency. In *Seminar on Concurrency*, pages 246–267, Springer-Verlag, 1984.

[Win84b] G. Winskel. Synchronization trees. *Theoretical Computer Science*, 34:33–82, 1984.

[Win86] G. Winskel. Event structures. In *Advanced Course on Petri Nets*, GMD, Bad Honnef, September 1986.