An Algebra for Probabilistic Processes

Vijay K. Garg[1]

Department of Electrical and Computer Engineering,
University of Texas, Austin, TX 78712-1084

Abstract

We define probabilistic languages and probabilistic automata over a finite set of events. We also define operators under which the set of probabilistic languages(p-languages) is closed, thus forming an algebra of p-languages. We show that this set is a complete partial order and our operators are continuous in it. Hence, recursive equations may be defined in this algebra using fixpoints of continuous functions. Thus, p-languages form a suitable theoretical foundation for specifying and analyzing probabilistic systems. The model can easily be extended for performance analysis by associating timing information with each event. We define many performance indices for systems expressed using this model and derive techniques to compute them.

1 Introduction

The theory for supervisory control of discrete event dynamical systems has been an active area of research since its initiation by Ramadge and Wonham [21, 22, 20]. Most research in this area has used theory of deterministic formal languages [2, 15, 14, 23, 12, 13]. However, discrete event dynamical systems that occur in practice generally have noise associated with them. Modeling these system require that probability be associated with any change in the system. The modeling techniques available for probabilistic systems are generally based on state-transition model such as Markov chains. In this paper, we present a simple algebraic probabilistic model that can be used for specification and performance analysis of dynamical systems. Our model is such that logical analysis, such as safety and progress properties, can be done in the same framework as analysis for performance and reliability.

We define probabilistic languages over a set of events. We show that the set of all p-languages forms an inf semi-lattice and a complete partial order with respect to a natural ordering of elements. Thus, recursive functions can be defined via fix-points on this set. We define various operators between probabilistic languages (automata) which can be used to build complex systems from simpler systems. In particular, we define regular operators (union, concatenation and concatenation closure) and show that these operators preserve finiteness, ordering and least upper bounds of chains. Thus, p-languages forms a suitable domain for specification and analysis of probabilistic discrete event dynamical systems. We define many performance indices for systems expressed using this model and derive algorithms to compute them. Our formalism has the advantage that it shows compositional properties. For example, we show in this paper that expected completion time of a system can be derived using expected completion time of its components.

An early attempt to generalize languages was done by researchers in fuzzy set theory [26] [25]. A fuzzy language is a fuzzy set defined on strings of events. Thus, every string has a grade of membership in $[0, 1]$. Our definition of probabilistic languages allows only those membership grades which satisfy certain consistency constraints. An advantage of these constraints is that the membership grade of a string can be viewed as the probability that the system executes that string. This results in a richer theory for systems that can be modeled by probabilistic languages.

[1]supported in part by grants from the University Research Institute, University of Texas at Austin, NSF Grant CCR-9110605, TRW award and IBM agreement 153. email: vijay@pine.ece.utexas.edu

There have also been attempts to generalize deterministic automata to probabilistic and stochastic automata [17][19] [4]. PCCS[6] also extends CCS[16] by explicitly introducing probability for the summation operator. These attempts emphasized generalization of state machines by attaching probabilities with edges, however there was little effort to generalize regular operations between languages. We have taken a more algebraic approach by emphasizing the operations defined on the set of probabilistic languages.

Our work has many things common with Markov chains, but there are two main differences. First, there is an emphasis on the alphabet which is generally missing from the theory of Markov chains. This alphabet has many implications on the questions we ask about a Markov chain. For example, we would be more interested in the sequence of events rather than the state after some number of events. Secondly, we do not require the probabilities of transitions from any state to sum up to 1. We assume that the system terminates at that state with the rest of the probability. This has implications in defining various operators between multiple Markov chains. For example, concatenation between two Markov chains implies that the second Markov chain begins executing whenever the first terminates.

Our work has many aspects common to CSP[9] and FRP[10]. In particular, we define the meaning of recursion similar to the definition in CSP (Also see [24]). The main difference arises because of probabilities associated with transitions.

This paper is organized as follows. Section 2 describes the frequently used notation in this paper. Section 3 defines probabilistic languages. Section 4 describes various operators defined between probabilistic languages. Section 5 defines probabilistic automata. Section 6 defines many measures of performance on systems defined using our formalism, and describes methods to compute them.

2 Notation

We use the following notation in this paper.

A	finite set of events
a, b	events
s, t, u, v	strings in A^*
K, L	p-languages, maps from A^* to $[0,1]$
$\mathcal{L}$	the set of p-languages
C, D	completion probability density function
f, g, h	language functions, maps from A^* to $[0,1]$
$\mathcal{F}$	the set of language functions
X, Y, Z	State Space
P, Q, R	State transition probability functions
$f \circ g$	convolution of f and g
$f^{(i)}$	$f \circ f \circ ... i\ times$
$f.g, fg$	product of f and g
f^i	$f.f....i\ times$
$K._e L$	concatenation binary operator in $\mathcal{L}$
$K +_e L$	choice binary operator in $\mathcal{L}$
$K + L$	addition operator in $\mathcal{R}$

Our precedence rules are given by the following list. All operators on the same line, such as $+$ and $-$ have equal precedence, and all operators on a given line have higher precedence than those on the lines below them.

exponentiation for . and $\circ$
$\circ$, .
Σ
$+, -$
$=, \leq, \geq$.
Thus, an expression such as $f \circ g + h$ means $(f \circ g) + h$ and not $f \circ (g + h)$.

We use calculational style of proofs for many of our Theorems. A proof that $[A \equiv C]$ will be rendered in our format as
A
= { hint why $[A \equiv B]$ }
B
= { hint why $[B \equiv C]$ }
C
We also allow implies ($\Rightarrow$) in the leftmost column. For a thorough treatment of this proof format we refer interested readers to [Dijkstra 90].

3 Probabilistic Languages

Definition 1 *A p-language L is defined as $L : A^* \rightarrow [0, 1]$ with the constraint that*
(C1) $L(\epsilon) = 1$
(C2) *for all s :* $\sum_{a \in A} L(sa) \leq L(s)$

The interpretation of $L(s)$ is taken as the probability that s occurs in the system. (C1) says that the null string is always possible in the system. (C2) captures the property that if a system executes sa, then it must have executed s. Thus, the combined probability $\sum_a L(sa)$ must not be greater that $L(s)$. (C2) also implies that if s is a prefix of t, then $L(s)$ is greater than or equal to $L(t)$.

This definition generalizes non-empty prefix closed languages. If L is required to map either to 0 or 1, and the second requirement says that $(L(sa) = 1) \Rightarrow (L(s) = 1)$, then we get the definition of a prefix-closed language.

We note that $L(s) \leq 1$ for all s can also be derived from (C1) and (C2), but we keep it explicitly in our formulation for simplicity. Based on L we can define a probability density function C_L on the set A^*.

$$C_L(s) = L(s) - \sum_a L(sa) \qquad \textit{for } s \in A^* \tag{1}$$

The interpretation of $C_L(s)$ is that it is the probability that the system does s and stops, that is it does not do anything afterwards. We will drop the subscript L, when clear from the context. As $\sum_a L(sa) \leq L(s)$, $C(s)$ is always greater than or equal to zero. Since $L(s)$ is less than or equal to 1, so is $C(s)$ for any s. Thus, $C(s)$ always lies between 0 and 1.

Example 2 Let L be defined as: $L(\epsilon) = 1, L(a) = 0.4, L(ab) = 0.2, L(b) = 0.6, L(ac) = 0.1$ and $L(s) = 0$ for all other strings s. Then it is easy to see that the L satisfies required properties. Hence it is a p-language. Further C for this p-language is: $C(a) = 0.1, C(ab) = 0.2, C(ac) = 0.1, C(b) = 0.6$

Example 3 The nil language I is defined as $I(\epsilon) = 1, I(s) = 0$ for $s \neq \epsilon$. For this language $C_I(\epsilon) = 1, C_I(s) = 0$ for $s \neq \epsilon$.

Example 4 Consider a Bernouli process. Every experiment has two outcomes a or b, with probability p and $(1 - p)$, respectively. Here, our alphabet $A = \{a, b\}$, and

$L(s) = p^{\#(a,s)}(1-p)^{\#(b,s)}$ for any $s \in A^*$, where $\#(a,s)$ represents the number of occurrences of a in the string s.
It is easy to check that this language satisfies (C1) and (C2). Also note that $C(s) = 0$ for any s.

We note that both p-languages and completion probability are functions defined from A^* to $[0,1]$. We call them language functions or simply functions. Let $\mathcal{F}$ be the set of all functions. We define a function $\Lambda : \mathcal{F} \to \mathcal{F}$ as follows:

$$\textit{For any } s, \qquad \Lambda(f)(s) = \sum_a f(sa)$$

Thus, if f is a p-language, then $\Lambda(f)$ gives the probability that the system will execute something after s. It is easy to check that Λ is a linear functional. With this notation, the constraint (C2) on f is
$\Lambda(f) \leq f$.
Similarly, for any p-language K, its corresponding completion pdf is easily derived as

$$C_K = K - \Lambda(K)$$

Now we make the following observations:

Lemma 5 *(a)* $\sum_s C(s) \leq 1$
(b) For any $s : L(s) = \sum_t C(st)$ *iff* $\lim_{k\to\infty} \sum_{|t|=k} L(st) = 0$
(c) $\sum_s C(s) = 1$ *iff* $\lim_{k\to\infty} \sum_{|t|=k} L(t) = 0$

Proof (a) : We first show that $\sum_{|s|=n} L(s) \leq 1$. This can be shown using induction on n. It is trivially true for $n = 0$. Assume that it is true for $n = k$. Then,
$\sum_{|s|=k+1} L(s)$
$= \{\ s = ta\ \}$
$\sum_{|t|=k} \sum_a L(ta)$
$\leq \{$ (C2) $\}$
$\sum_{|t|=k} L(t)$
$\leq\{$ Induction hypothesis $\}$
1
We now show that $\sum_{s\in A^*} C(s) \leq 1$
Let

$$S(n) = \sum_{|s|\leq n} C(s)$$

Then $\sum_s C(s) = \lim_{n\to\infty} S(n)$ It is easy to show by induction that $S(n) = 1 - \sum_{|s|=n+1} L(s)$
Therefore, $0 \leq S(n) \leq 1$. Since $\{S(n)\}$ is a monotone increasing sequence bounded above by 1, the limit exists and is at most 1.

Proof (b): From equation (1), we get that

$$L(s) = C(s) + \sum_a L(sa)$$

Therefore by repeated application of this equation we get,
$L(s) = \sum_{|t|<k} C(st) + \sum_{|t|=k} L(st)$ *for all* $k \geq 1$
$= \lim_{k\to\infty} \sum_{|t|<k} C(st) + \lim_{k\to\infty} \sum_{|t|=k} L(st)$

$= \sum_t C(st) + \lim_{k \to \infty} \sum_{|t|=k} L(st)$
The lemma follows.

Proof (c): By substituting ϵ for s in part (b). $\square$

The first part of the above proposition justifies the use of probability density function for completion. The second part of the above lemma indicates that given any pdf, we can also view it as completion probability of a system. It also suggests that, we could have alternatively defined p-languages using C, and defined L in terms of C. This, however, required that $\sum C(s) = 1$.

We call a system terminating if $\sum C(s) = 1$. From now on we will assume that all our systems are terminating. This assumption is only for simplicity, as the theory we develop can be easily extended to the case when there is a non-zero probability that the system does not terminate.

3.1 An Order on P-languages

We can now define an ordering between p-languages. We use $\mathcal{L}_{\mathcal{A}}$(or simply $\mathcal{L}$) to denote the set of all p-languages defined over the set A.

Definition 6 *Let $K, L \in \mathcal{L}$. Then,*
$K \preceq L$ if and only if $\forall s : K(s) \leq L(s)$

Example 7 Let $K, L, U \in \mathcal{L}$ be defined as : $K(a) = 0.4, K(ab) = 0.3$
$L(a) = 0.5, L(b) = 0.4, L(ab) = 0.3$
$U(a) = 0.4, U(b) = 0.5, U(ab) = 0.4$
Then, $K \preceq L$, and $K \preceq U$, while L and U are incomparable.

Lemmas 8 and 10 describe the properties of this order.

Lemma 8 *$(\mathcal{L}, \preceq)$ is an* inf *semi-lattice.*

Proof: Let $K, L \in \mathcal{L}$. Then their least upper bound (if it exists) is denoted by $K \sqcup L$, and the greatest lower bound by $K \sqcap L$. They can be obtained as:
$K \sqcup L(s) = \sup(K(s), L(s))$
$K \sqcap L(s) = \inf(K(s), L(s))$
Let $K, L \in \mathcal{L}$ and $V = K \sqcap L$. It is easy to see that V is also a function from A^* to $[0, 1]$, and that $V(\epsilon) = 1$. We now show that
for all s : $\sum_a V(sa) \leq V(s)$.
For any s
$\sum_a V(sa)$
$=$ {definition $\sqcap$ }
$\sum_a \inf(K(sa), L(sa))$
$\leq$ { arithmetic, using induction }
$inf(\sum_a K(sa), \sum_a L(sa))$
$\leq$ { $K, L \in \mathcal{L}$ }
$inf(K(s), L(s))$
$=$ {definition $\sqcap$ }
$V(s)$. $\square$

In example 7, $V = L \sqcap U$ is defined as $V(a) = 0.4, V(b) = 0.4, V(ab) = 0.3$. However, $K \sqcup L$ may not exist for all K, L. As an example consider the following: $K(\epsilon) = 1, K(a) = 0.6$ $L(\epsilon) = 1, L(b) =$

0.7 Then $K \sqcup L$ is not defined (does not belong to $\mathcal{L}$). Even though $\sqcup$ may not exist for any set of p-languages, it exists for any chain of p-languages. The definition of chain and the Lemma is given below.

Definition 9 *We call a family of p-languages $\{L_i | i = 0, 1..\}$ to be a chain iff $L_i \preceq L_{i+1}$ for all i.*

Lemma 10 *$\preceq$ is a complete partial order(cpo) on the set $\mathcal{L}$. I is the least element in this cpo.*

Proof: It is easy to verify that $\preceq$ is reflexive, antisymmetric and transitive. Thus, it is a partial order. We just need to show that if L_i is a chain of p-languages, then its least upper bound exists. We define $\bigsqcup L_i(s) = \sup_i L_i(s)$. This is well defined because $L_i(s)$ is a monotonic sequence bounded above by 1. We also write it as $\lim_{n\to\infty} L_i(s)$. It is easy to see that $\bigsqcup L_i$ is also a function from A^* to $[0, 1]$, and that $\bigsqcup L_i(\epsilon) = 1$. We now show that
for all s : $\quad \sum_a \bigsqcup L_i(sa) \leq \bigsqcup L_i(s)$.
For any s,
$\quad \sum_a \bigsqcup L_i(sa)$
$=$ {definition $\sqcup$ }
$\quad \sum_a \sup_i L_i(sa))$
$=$ { L_i monotone }
$\quad \sum_a \lim_{i\to\infty} L_i(sa))$
$=$ { finite sum }
$\quad \lim_{i\to\infty} \sum_a L_i(sa))$
$\leq$ { $L_i \in \mathcal{L}$ }
$\quad \lim_{i\to\infty} L_i(s))$
$=$ {definition $\bigsqcup$ }
$\quad \bigsqcup L_i(s)$.

As a result of the above Lemma, we can easily compute fixed points of continuous functions. An alternative way of defining fixed points via Cauchy sequences is given in Appendix A.

4 Operators between p-languages

We now define some operations between various languages. These operators are useful in describing a complex system as a combination of many simple systems.

4.1 choice

This operator captures non-deterministic choice between two systems. Given two p-languages L_1 and L_2, and a real number e between 0 and 1, the combined system is denoted by $L_1 +_e L_2$ for e in $[0, 1]$. We use e' to be equal to $1 - e$ in rest of the paper.

Definition 11 $L_1 +_e L_2 = eL_1 + e'L_2$

The interpretation of the above definition is : do L_1 with probability e or L_2 with probability $1 - e$. It is easy to verify that the p-language defined above satisfies constraints (C1) and (C2).[2]

[2]This operator can easily be generalized for multiple arguments. In essence, the choice operator represents a convex combination of p-languages

We now derive an expression for the completion probability density function (pdf) for the composed system.
$C_1 +_e C_2 = L_1 +_e L_2 - \Lambda(L_1 +_e L_2)$
$= eL_1 - \Lambda(eL_1) + e'L_2 - \Lambda(e'L_2)$
$= eC_1 + e'C_2$

The following proposition describes property of choice with respect to the ordering.

Definition 12 A function $f : \mathcal{L} \rightarrow \mathcal{L}$ is called *monotone* iff

$$K \preceq L \Rightarrow f(K) \preceq f(L)$$

It is called *continuous* iff for all chains $\{L_i\}$,

$$f(\bigsqcup_i L_i) = \bigsqcup_i f(L_i)$$

It can be easily shown that every continuous operator is also monotone.

Proposition 13 *Choice is a continuous operator in both of its arguments.*

Proof: We need to show that for any e,

$$K +_e (\bigsqcup_i L_i) = \bigsqcup_i (K +_e L_i)$$

The above is easily verified using definitions.

4.2 Concatenation

This operator captures sequencing of two systems. Given two p-languages L_1 and L_2, and a real number e between 0 and 1, the combined system is denoted by $L_1 ._e L_2(s)$ for e in $[0, 1]$.

Definition 14 $L_1 ._e L_2(s) = L_1(s) + e\sum_{t<s} C_1(t)L_2(s/t)$
$= L_1(s) - eC_1(s) + e\sum_t C_1(t)L_2(s/t)$

The above definition has the following interpretation. The resulting system does L_1, and then on completion does L_2 with probability e. The probability that s occurs in the composed system is equal to the probability that it occurs in the first system, or a part of it occurs in the first system and a non-null part in the second. It can be shown that $\mathcal{L}$ is closed under the operation of $._e$. It is easy to check that $L_1 ._e L_2(\epsilon) = L_1(\epsilon) - eC_1(\epsilon) + eC_1(\epsilon)L_2(\epsilon) = 1$. Therefore, (C1) holds. We later derive an expression for $C_1 ._e C_2 = L_1 ._e L_2 - \Lambda(L_1 ._e L_2)$, and show that it is always positive. Hence (C2) also holds.

We define the convolution operators between two real-valued functions on A^* as follows:

$$f \circ g(s) = \sum_t f(t)g(s/t)$$

Thus $L_1 ._e L_2(s)$ is defined more simply as $L_1(s) - eC_1(s) + eC_1 \circ L_2(s)$
The following proposition describes certain properties of the convolution operator.

Proposition 15 *(a) Convolution is associative, i.e.*

$$f \circ (g \circ h) = (f \circ g) \circ h$$

(b) I is the identity for convolution, i.e.

$$f \circ I = I \circ f = f$$

(c) Convolution is continuous in both arguments, i.e.

$$f \circ (\bigsqcup_i g_i) = \bigsqcup_i (f \circ g_i), (\bigsqcup_i f_i) \circ g_i = \bigsqcup_i (f \circ g_i)$$

(d) Convolution is a linear operator, i.e.

$$f \circ (\alpha g + h) = \alpha f \circ g + f \circ h$$

(e) If $g(\epsilon) = 1$

$$\Lambda(f \circ g) = f \circ \Lambda(g) + \Lambda(f)$$

Proof:
(a)For any s, $f \circ (g \circ h)(s)$
$= \sum_t f(t) g \circ h(s/t)$
$= \sum_t f(t) \sum_u g(u) h(s/tu)$
$= \sum_u \sum_t f(t) g(u) h(s/tu)$
$= \sum_u \sum_t f(t) g(tu/t) h(s/tu)$ { v = tu }
$= \sum_u f \circ g(v) h(s/v)$
$= \sum_v f \circ g(v) h(s/v)$
$= (f \circ g) \circ h(s)$

(b) By substituting I in the expression for convolution.

(c) We need to show that $f \circ \bigsqcup_i g_i = \bigsqcup_i f \circ g_i$. For any s,
$f \circ \bigsqcup_i g_i(s)$
={definition $\circ$ }
$\sum_t f(t)(\bigsqcup_i g_i(s/t))$
= { real analysis }
$\bigsqcup_i \sum_t f(t) g_i(s/t)$
={definition $\circ$ }
$\bigsqcup_i f \circ g_i$
The continuity in the other argument is similarly proved.

(d) For any s,
$f \circ (\alpha g + h)(s)$
$= \sum_t f(t)(\alpha g + h)(s/t)$
$= \sum_t f(t)(\alpha g(s/t) + h(s/t))$
$= \sum_t f(t) \alpha g(s/t) + \sum_t f(t) h(s/t))$
$= \alpha f \circ g + f \circ h$

(e) For any s,
$\Lambda(f \circ g)$
$= \sum_a f \circ g(sa)$

$= \sum_a \sum_{t<=sa} f(t)g(sa/t)$
$= \sum_a \sum_{t\leq s} f(t)g(sa/t) + \sum_a f(sa)g(\epsilon)$
$= f \circ \Lambda(g) + \Lambda(f)$

We use $f^{(2)}$ to represent $f \circ f$. In general we use $f^{(i)}$ to represent i times convolution of f with itself. This is well-defined because of associativity of $\circ$. We define $f^{(0)}$ to be the I function. From the proof of associativity, we also note that $\sum_{s_i, 0\leq i\leq n-1} C(s_0)C(s_1)...C(s_{n-1}) = C^{(n)}(s)$. We show the following proposition.

Proposition 16 *Let $\sum_s C(s) = 1$. Then, $\sum_s C^{(n)}(s) = 1$ for all n.*

Proof: We use induction on n. The proposition is clearly true for $n = 1$. Assume that it is true for $n = k$. Then, $\sum_s C^{(k+1)}(s)$
$= \sum_s C^{(k)} \circ C(s)$
$= \sum_s \sum_t C^{(k)}(t)C(s/t)$
$= \sum_u \sum_t C^{(k)}(t)C(u)$
$= \sum_u C(u) \sum_t C^{(k)}(t)$
$= 1$

We now derive the expression for completion probability for concatenation.
$C_1 ._e C_2 = L_1.L_2 - \Lambda(L_1.L_2)$
$=$ { definition of $L_1.L_2$}
$L_1 + eC_1 \circ L_2 - eC_1 - \Lambda(L_1 - eC_1 + eC_1 \circ L_2)$
$=$ {definition $C_1 = L_1 - \Lambda(L_1)$}
$C_1 + e(C_1 \circ L_2 - C_1 - \Lambda(C_1 \circ L_2) + \Lambda(C_1))$
$=$ { property convolution }
$C_1 + e(C_1 \circ L_2 - C_1 - C_1 \circ \Lambda(L_2))$
$=$ { property convolution, definition C_2 }
$C_1(s) + eC_1 \circ C_2(s) - eC_1(s)$
$=$ { definition e'}
$e'C_1(s) + eC_1 \circ C_2(s)$

We note that if $e = 0$, then the system is identical as before. On the other hand if $e = 1$, then the resulting system is obtained as a pure convolution. Since convolution is continuous, it is easy to show the following Theorem.

Proposition 17 *Concatenation is continuous in its second argument, i.e.*

$$K ._e \bigsqcup L_i = \bigsqcup K ._e L_i$$

Proof: $K ._e \bigsqcup L_i$
$= K - eC + eC \circ \bigsqcup L_i$
$= \bigsqcup K - eC + eC \circ L_i$
$= \bigsqcup K ._e L_i$ $\square$

However, concatenation is not even monotone in its first argument as shown by the following example. Let $K(a) = 0.2$, and $L(b) = 1.0$. Now $I \preceq K$, but $I ._1 L$ is not comparable to $K ._1 L$.

4.3 Recursion

For any $K \in \mathcal{L}$, we can use $+_e K$ and $._e K$ as unary operators defined from $\mathcal{L}$ to itself. For any $X \in \mathcal{L}$, $+_e K(X) = K +_e X$, and $._e K(X) = K ._e X$. As shown earlier, both of these operators are continuous

in the cpo of $\mathcal{L}$. Note that we use concatenation only in the second argument as it is not continuous for the first argument. Therefore, any composition of these operators will also be continuous. We can also define recursion operator denoted by $\mu X.F(x)$, where F is any function built out of $+$ and .. We define $\mu X.F(X) = \bigsqcup F^i(I)$. We show that the above p-language is a fixed-point of F, i.e. it satisfies $X = F(X)$.
$X = \bigsqcup F^i(I)$
$= \{$ application of F $\}$
$F(X) = F(\bigsqcup_{i\geq 0} F^i(I))$
$= \{$ F continuous, $\{F^i(I)\}$ a chain $\}$
$\bigsqcup_{i\geq 0} F^{i+1}(I)$
$= \{$ I minimum element $\}$
$\bigsqcup_{i\geq 0} F^i(I)$

Proposition 18 *Let $\{X_i\}$ be a family of p-languages defined by $X_{n+1} = A + B.X_n$, where $+$ is simple pointwise addition in $\mathcal{R}$. Then, $X_n = B^n.X_0 + \sum_{i\leq n-1} B^i.A$*
If . is replaced by $\circ$ uniformly, then the result holds.

Proof: Using induction on n.

Example 19 Let us consider the p-language defined by the following equation:
$X = K +_e X$. Then $X = \bigsqcup(K+_e)^i(I)$. We define $X_n = \bigsqcup_{i\leq n}(K+_e)^i(I)$. We list first few X_n's.
$X_0 = I$
$X_1 = K +_e I$
$X_2 = K +_e (K +_e I)$
$X_{j+1} = K +_e X_j$
$X_{j+1} = eK + e'X_j$
Thus, $X_j = (eK)^j + \sum_{i\leq n-1} e'^i eK$ (using Proposition 18). Then, $X_* = \lim_{j\to\infty} X_j$ We first consider the case when $e = 1$. Then all X_j except X_0 are K. Therefore, $X_* = K$. When $e = 0$, then $X_* = I$. When $0 < e < 1$, the above expression in limit reduces to
$X_* = \Sigma e'^i eK$
$= (1/1 - e')eK$
$= (1/e)eK = K$

Example 20 Consider the p-language defined by the following equation:
$X = K._eX$. If $e = 0$, then $X = K$. We now assume that $e \geq 0$.
Then $X = \bigsqcup(K._e)^i(I)$. We define $X_n = \bigsqcup_{i\leq n}(K._e)^i(I)$. We list first few X_n's.
$X_0 = I$
$X_1 = K._eI$
$X_2 = K._e(K._eI)$
$X_{j+1} = K._eX_j$
$X_{j+1} = K - eC + eC \circ X_j$
We use the following notation:

$$D(l,u) = \sum_{i=l}^{i=u} e^i C^{(i)}$$

$$D(j) = \sum_{i\geq j} e^i C^{(i)}$$

Thus, $X_j = e^j C^j + D(0, j-1) \circ (K - eC)$
$= e^j C^j + D(0, j-1) \circ K - D(1, j)$

$= D(0, j-1) \circ K - D(1, j-1)$
Thus, $X^* = D(0) \circ K - D(1)$

We will use this as the definition of repeated concatenation in the next section.

4.4 Concatenation Closure

This operator captures repetition of any system. Given any p-language L and a real number e between 0 and 1, the combined system is denoted by $L^{*_e}(s)$ for e in $[0, 1]$.

Definition 21 $L^{*_e}(s) = \sum_n e^n C^{(n)} \circ L(s) - \sum_{i \geq 1} e^i C^{(i)}(s)$
$= D(0) \circ L - D(1)$

The interpretation of the above definition is - do L and then repeat it with probability e.

$C^{*_e} = L^{*_e} - \Lambda(L^{*_e})$
$=$ { definition of L^{*_e} }
$D(0) \circ L - D(1) - \Lambda(D(0) \circ L - D(1))$
$=$ { property of Λ }
$D(0) \circ L - D(1) - D(0) \circ \Lambda(L) - \Lambda(D(0)) + \Lambda(D(1))$
$=$ { $\Lambda(I)$ is zero uniformly }
$D(0) \circ L - D(1) - D(0) \circ \Lambda(L)$
$=$ { $C = L - \Lambda(L)$ }
$D(0) \circ C - D(1)$
$=$ { dividing and multiplying the first term by e }
$e^{-1} D(1) - D(1)$
$=$ { algebra }
$e'/eD(1)$
$=$ { definition D }
$e'/e \sum_{n \geq 1} e^n C^{(n)}$

Example 22 Consider a process of tossing a coin which is repeated till the result of the toss is a head. Assuming that the tail comes with probability e, this system can be modeled as follows:
$A = \{toss\}$
$L(\epsilon) = 1$, $L(toss) = 1$, $L(s) = 0$ for other s.
Then, our system is just repeated concatenation of L. For the primitive function, completion pdf is given by $C(toss) = 1, C(s) = 0$ for other s. By using formulas for convolution, we obtain that $C^{(k)}((s) = 1$ only for $s = toss.toss.toss...k\ times$. Thus, $L^{*_e}(toss^k) = e^{k-1} + e^k - e^k = e^{k-1}$ for $k \geq 1$ by definition 21. This expression can be verified by calculating it directly from the problem. Similarly, $C^{*_e}(toss^k) = e'/e * e^k = e^{k-1}e'$ for $k \geq 1$.

4.5 Automata for p-languages

Many times it is easier to describe a p-language using an automata associated with it. We define a p-automata M over the alphabet set A as follows:
(X, x_0, P)
X: a set of states

x_0: initial state
$P : X \times A \times X \to [0,1]$ with the constraint that: $\sum_a \sum_j P(x_i, a, x_j) \leq 1$ for all i.

The interpretation of the above automata is as follows. If the system is in state x_i, then it makes a transition to x_j on a with probability $P(x_i, a, x_j)$. We also define

$$S(x) = 1 - \sum_{a \in A} \sum_{y \in X} P(x, a, y)$$

If the system is in states x, then it terminates with probability S(x).

We first compute the probability of a string s that occurs in a probabilistic automata. We note that it may be non-deterministic, and there may be multiple paths corresponding to the same string. Given any directed path in the above automata starting from x_0, the probability that path occurs is simply the product of probabilities of individual edges on that path. We then define a p-language L for the machine M as follows:
$L(s)$ = sum of all paths that trace s in M.

Proposition 23 *L is a p-language.*

Proof: If each edge is labeled from [0,1], it is easy to see that $L(s) \geq 0$ for all s. Since ϵ traces only one path - null path- $L(\epsilon)$ is 1 as required. We just need to show that $\sum_a L(sa) \leq L(s)$. $\sum_a L(sa)$ is equal to sum of probabilities of all paths that trace sa. Let U be this set of paths, and V be the projection of these paths on string s. Consider any path v in V. For each path v in V there exists a set of paths U_v in U. Let π be the function that gives probability for any path. Then for any s,
$\sum_a L(sa)$
$= \sum_{u \in U} \pi(u)$
$= \sum_{v \in V} \sum_{u \in U_v} \pi(u)$
$= \sum_{v \in V} \sum_a \pi(va)$
$\leq \sum_{v \in V} \pi(v)$
$= L(s)$ □

Thus, every p-automata defines a p-language. We now show that for every p-language there exists a p-automata. The p-automata for any p-language L is defined as (X, x_0, P):
$X = A^*$
$x_0 = \epsilon$
$p(x_i, a, x_j) = L(x_j)/L(x_i) \ if \ x_j = x_i a$
$= 0 \ otherwise.$

In this machine there exists a unique path for each string. The probability of that path is given by the product of probability of individual edges. By our construction, $P(x_j)$ will be equal to $L(x_j)$.

A special interesting class of p-automata is that containing finite number of states. The characterization of p-languages of finite p-automata is given using the relation defined as follows:

Definition 24 *$s \equiv t$ iff $\forall x : L(sx)/L(s) = L(tx)/L(t)$*

This relation is clearly reflexive, symmetric and transitive. If the number of equivalence classes of this relation is finite then we say that the p-language L has finite index. Now we can show the following Theorem:

Proposition 25 *A p-language L has finite index iff there exists a finite deterministic p-automata.*

Proof: If L has finite index, we construct a p-automata with states as equivalence classes. If there exists a deterministic finite p-automata, the equivalence classes then each state defines an equivalence class. □

The p-automata constructed in the above Lemma is the one with minimum number of states. Given any p-automata, its minimization can be done in a manner similar to the algorithm used for deterministic finite state machines.

If p-languages are given to us as p-automata, then the operations of choice, concatenation, and concatenation closure can be realized easily.

Proposition 26 *If L_1 and L_2 have finite indices, then $L_1 +_e L_2$, $L_1 ._e L_2$, and L_1^{*e} also have finite indices.*

Proof: Let $M_1 = (X, x_0, P)$, $\quad M_2 = (Y, y_0, Q)$

Case 1: $L_1 +_e L_2$
We define $M_1 +_e M_2 = (X \cup Y \cup \{z_0\}, z_0, R)$ where
z_0 is an additional state
$R(z_0, x, a) = eP(x_0, x, a)$
$R(z_0, y, a) = e'Q(y_0, y, a)$
$R(x_i, x_j, a) = P(x_i, x_j, a)$ for all $x_i \in X, x_j \in X, a \in A$
$R(y_i, y_j, a) = Q(y_i, y_j, a)$ for all $y_i \in Y, y_j \in Y, a \in A$
It is easy to see that R satisfies constraint on p-automata. For any path in M_1 from x_0, there exists a path in the new machine with an extra factor of e. For paths in M_2 there is a factor of e'. Thus, $L(M_1 +_e M_2) = L(M_1) +_e L(M_2)$

Case 2: $L_1 ._e L_2$
$M_1 ._e M2 = (X \cup Y, x_0, R)$
$R(x, y, a) = eS_1(x)Q(y_0, y, a)$ for all $x \in X, y \in Y, a \in A$
$R(x_i, x_j, a) = P(x_i, x_j, a)$ for all $x_i \in X, x_j \in X, a \in A$
$R(y_i, y_j, a) = Q(y_i, y_j, a)$ for all $y_i \in Y, y_j \in Y, a \in A$
It is easy to verify that the definition is consistent, i.e.
$L(M_1 ._e M_2) = L(M_1) ._e L(M_2)$
Consider any string s. All the paths in M_1 are also in $M_1 ._e M_2$ - therefore, the term of $L_1(s)$. Other possible paths are through transitions with probability e. The factor of $eS(x)$ is multiplied to any of these paths. The path in the second machine is always non-null therefore the case when $s/t = \epsilon$ is subtracted.

Case 3: L_1^{*e}
$M_1^{*e} = (X, x_0, R)$
$R(x, y, a) = eS(x)P(x_0, y, a) + P(x, y, a)$
We leave it to readers to verify that $L(M^*) = L(M)^*$

□

The above Lemma can be viewed as a generalization of Kleene's theorem in one direction.

5 Performance Indices

In this section, we describe how many performance indices of a complex system can be derived from performance indices of its components.

5.1 Average completion time

Let $T : A^* \to \mathcal{R}^+$ be any function that gives us time spent on execution of the string s. We require following properties of T.
(1) $T(\epsilon) = 0$
(2) $T(st) = T(s) + T(t)$
In the following discussion, we will use only these two properties of T. Thus, our discussion is valid for any other performance index that satisfies these two properties. We define average completion time as follows:

$$E[C,T] = \sum_s C(s)T(s)$$

We need to derive some more properties of convolution operator before discussing these properties for other operators.

Proposition 27 *(a)* $E[C_1 \circ C_2, T] = E[C_1, T] + E[C_2, T]$
(b) $E[C^{(n)}, T] = nE[C, T]$

Proof (a): $E[C_1 \circ C_2, T]$
$=$ {definition expectation}
$\sum_s C_1 \circ C_2(s)T(s)$
$=$ {definition convolution, breaking s}
$\sum_s \sum_t C_1(t)C_2(s/t)T(t.s/t)$
$=$ {property of T}
$\sum_u \sum_t C_1(t)C_2(u)(T(t) + T(u))$
$=$ {all operators linear}
$\sum_u \sum_t C_1(t)C_2(u)T(t) + \sum_u \sum_t C_1(t)C_2(u)T(u))$
$=$ {rearranging terms}
$(\sum_u C_2(u)) \sum_t C_1(t)T(t) + (\sum_t C_1(t)) \sum_u C_2(u))T(u))$
$=$ {assumption of terminating systems}
$\sum_t C_1(t)T(t) + \sum_u C_2(u)T(u))$
$=$ {definition expectation}
$E[C_1, T] + E[C_2, T]$

Proof (b): We use induction on n. The proposition is trivially true for $n = 1$. Assume that it is true for $n = k$.
$\sum_s C^{(k+1)}T(s)$
$= \sum_s C^{(k)} \circ C(s)T(s)$
$= \sum_s C^{(k)}T(s) + \sum_s C(s)T(s)$
$= (k+1) \sum_s C(s)T(s)$

Proposition 28 $E[C_1 +_e C_2, T] = eE[C_1, T] + e'E[C_2, T]$
$E[C_1 \cdot_e C_2] = E[C_1, T] + eE[C_2, T]$
$E[C^{*_e}, T] = E[C, T]/e'$

Proof (a) :
We use that $C = eC_1 + e'C_2$

Proof (b) :
$E[C_1 \cdot_e C_2]$

$= E[e'C_1 + eC_1 \circ C_2, T]$
$= e'E[C_1, T] + eE[C_1 \circ C_2, T]$
$= e'E[C_1, T] + eE[C_1, T] + eE[C_2, T]$
$= E[C_1, T] + eE[C_2, T]$

Proof (c) :
$E[C^*, T]$
$= E[e'/e \sum_{i \geq 1} e^i C^{(i)}, T]$
$= e'/e \sum_{i \geq 1} e^i E[C^{(i)}, T]$
$= e'/e \sum_{i \geq 1} e^i i E[C, T]$
$= e'/eE[C, T] \sum_{i \geq 1} e^i i$
$= E[C, T]/e'$

The above Theorem shows that average completion time can easily be calculated in a modular manner. This fact can be used for analysis of timed system [3] [11] [18].

5.2 Average Reliability

In this section, we assume that we are given a reliability function $R : A \rightarrow [0, 1]$ for each event in A. We interpret $R(a)$ as the probability that a can be executed without failure. We require following properties of R.
(1) $R(\epsilon) = 1$
(2) $R(st) = R(s)R(t)$
In the following discussion, we will use only these two properties of R. Thus, our discussion is valid for any other performance index that satisfies these two properties. We define average reliability as follows: $E[C, R] = \sum_s C(s)R(s)$

We need to derive some more properties of convolution operator before discussing these properties for other operators.

Proposition 29 *(a)* $E[C_1 \circ C_2, R] = E[C_1, R]E[C_2, R]$
(b) $E[C^{(n)}, R] = (E[C, R])^n$

Proof (a): $E[C_1 \circ C_2, R]$
$= \{$definition expectation$\}$
$\sum_s C_1 \circ C_2(s)R(s)$
$= \{$definition convolution, breaking $s\}$
$\sum_s \sum_t C_1(t)C_2(s/t)R(t.s/t)$
$= \{$property of R$\}$
$\sum_u \sum_t C_1(t)C_2(u)R(t)R(u)$
$= \{$rearranging terms$\}$
$\sum_u \sum_t C_1(t)C_2(u)R(t). \sum_u \sum_t C_1(t)C_2(u)R(u)$
$= \{$rearranging terms$\}$
$(\sum_u C_2(u) \sum_t C_1(t)R(t).(\sum_t C_1(t) \sum_u C_2(u)R(u)$
$= \{$assumption of terminating systems$\}$
$\sum_t C_1(t)R(t). \sum_u C_2(u)R(u)$
$= \{$definition expectation$\}$
$E[C_1, R]E[C_2, R]$

Proof (b): From part (a) using induction on n.

Proposition 30 *(a)* $E[C_1 +_e C_2, R] = E[C_1, R] + e'E[C_2, R]$
(b) $E[C_1.eC_2, R] = e'E[C_1, R] + eE[C_1, R]E[C_2, R]$
(c) $E[C^*, R] = e'E[C, R]/(1 - eE[C, R])$

Proof(a): We use that $C = eC_1 + e'C_2$.

Proof (b) :
$E[C_1.eC_2, R]$
$= E[e'C_1 + eC_1 \circ C_2, R]$
$= e'E[C_1, R] + eE[C_1 \circ C_2, R]$
$= e'E[C_1, R] + eE[C_1, R]E[C_2, R]$

Proof (c) : $E[C^*, R]$
$= E[e'/e \sum_{i \geq 1} e^i C^{(i)}, R]$
$= e'/e \sum_{i \geq 1} e^i E[C^{(i)}, R]$
$= e'/e \sum_{i \geq 1} (eE[C, R])^i$
$= e'E[C, R]/(1 - eE[C, R])$

The above proposition has intuitive interpretation. When two systems are concatenated with probability e, then with probability e' the resulting reliability is the reliability for the first system, and with e it is the product. When operations of a system are repeated we get the expression $e'E[C, R]/(1 - eE[C, R])$. Thus, if $e = 0$, we get $E[C, R]$. If $E[C, R] = 1$, then we get 1 for all e, and if $e = 1$, we get the reliability 0 for $E[C, R] < 1$.

6 Conclusions

In this paper we have described a modular formalism for specification and analysis of probabilistic discrete event dynamical system. This formalism is quite natural to apply to non-deterministic and stochastic systems. We have shown that systems defined in our formalism form a complete partial order under a suitable ordering relation. As a consequence, it is easy to define systems using recursion. We show that performance indices such as average completion time and average reliability can be computed in a modular manner using operators in our algebra.

There are many interesting future research directions. [7] describes concurrent regular expressions which are extensions of regular expressions for concurrent systems. It will be interesting to define a probabilistic analogue to that.

7 Acknowledgements

I am thankful to Professors H. Cragon, R. Kumar, G. Sasaki, and E.Swartzlander Jr. for some useful discussions on this topic. I am especially grateful to Ratnesh Kumar for pointing out that the definition of the concatenation operation is very similar to convolution. He also pointed out an error in Proposition 25 of an earlier draft. This research was supported in part by grants from the Bureau of Engineering Research and University Research Institute, University of Texas at Austin, and NSF Grant CCR-9110605, Ernest Cockerel Chair in Engineering (held by Prof. Harvey Cragon), and Schlumberger Chair in Engineering(held by Prof. Earl Swartzlander Jr.).

References

[1] R. D. Brandt, V. Garg, R.Kumar, F. Lin, S. I. Marcus, W. M. Wonham, "Formulas For Calculating Supremal Controllable and Normal Sublanguages," *Systems and Control Letters*, vol. 15, pp. 111-117, Aug. 1990.

[2] R. Cieslak, C. Desclaux, A. S. Fawaz and P. Varaiya, "Supervisory Control of Discrete-Event Processes with Partial Observations," *IEEE Trans. A. C.* , vol. 33, no. 3, pp. 249-260, 1988.

[3] V. Carchiolo, A. Faro, M. Malgeri "A Tool for the Performance Analysis of Concurrent Systems," Proc. BCS-FACS Workshop on Specification and Verification of Concurrent Systems, Scotland, July 1988, published by Springer-Verlag 1990, pp 121-139.

[4] E. Doberkat, *Stochastic Automata: Stability, Nondeterminism and Prediction*, Lecture Notes in Computer Science 113, Springer-Verlag 1981.

[5] W. Feller, An Introduction to Probability Theory and Its Applications, Wiley, New York, 1970.

[6] A. Giacalone, C. Jou, S.A. Smolka, "Algebraic Reasoning for Probabilistic Concurrent Systems", *Proc. Programming Concepts and Methods* M. Broy, & C.B.Jones (editors), Elsevier Science Publishers B.V. (North-Holland), IFIP 1990

[7] V. Garg, "Modeling of Distributed Systems by Concurrent Regular Expressions", *Proc. 2nd International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols,* Vancouver, Dec 1989.

[8] M. Heymann, "Concurrency and Discrete Event Control," *IEEE Control Systems Magazine*, vol. 10, no. 4, pp. 103-112, 1990.

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

[10] K. Inan, P. Varaiya, "Finitely recursive process models for discrete event systems," *IEEE Trans. Autom. COntrol*, vol. 33, no. 7, pp. 626-639, July 1988.

[11] F. Jahanian, A. Mok "Safety Analysis of Timing Properties in Real-Time Systems," *IEEE Trans. on Software Engineering*, SE-12(9), pp. 890-904, 1986.

[12] R. Kumar, V. K. Garg, S.I. Marcus, "Language Stability and Stabilizability of Discrete Event Dynamic Systems," accepted for *SIAM Journal on Control and Optimization*

[13] R. Kumar, V. K. Garg, S.I. Marcus, "On ω-Controllability and ω-Observability of Discrete Event Dynamic Systems," *to appear IEEE Transactions on Automatic Control*, Sept. 1992.

[14] S. LaFortune,"Modelling and Analysis of Transaction Execution in Database Systems," *IEEE Trans. Auto. Control*, vol. 33, no. 5, pp. 439-447, May 1988.

[15] F. Lin and W. M. Wonham, "On Observability of Discrete-Event Systems," *Information Sciences*, vol. 44, no. 3, pp. 173-198, 1988.

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

[17] Azaria Paz *Introduction to Probabilistic Automata* Academic Press, 1971.

[18] J. Quemada, A. Azcoraa, D. Frutos, "TIC: A Timed Calculus for LOTOS," Formal Description Techniques II, S.T. Vuong (Editor), Elsevier Science Publishers B.V. (North-Holland) IFIP, 1990, pp 195-209.

[19] M.O.Rabin, "Probabilistic Automata," *Information and Control*, Vol. 6, pp 230-245, 1963.

[20] P.J. Ramadge and W.M. Wonham, "The control of discrete event systems," *Proc. of the IEEE*, vol. 77, no. 1, pp 81-98, Jan 1989.

[21] P. J. Ramadge and W. M. Wonham, "Supervisory Control of a Class of Discrete Event Procesess," *SIAM J. Control and Optim.* , vol. 25, pp. 206–230, 1987.

[22] P. J. Ramadge and W. M. Wonham, "On the Supremal Controllable Sublanguage of a given Language," *SIAM J. Control and Optim.* , vol. 25, pp. 637–659, 1987.

[23] R. Smedinga,"Using trace theory to model Discrete Event Systems," *Discrete Event Systems: Models and Applications*, Lecture Notes in Control and Information Sciences, vol. 103, Springer Verlag, pp 81-99.

[24] J.E.Stoy, "Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory," *MIT Press,* 1977.

[25] L.A.Zadeh, "Fuzzy Sets," *Information and Control* Vol. 8, June 1965, pp 338-353.

[26] E.T. Lee, and L.A.Zadeh, "Note on Fuzzy Languages," *Information Sciences 1* 1969, pp 421-434.

8 Appendix A

In this appendix we show that the set of p-languages form a metric space.

Definition 31 *$d : \mathcal{L} \times \mathcal{L} \to \mathcal{R}$ is defined as:*
$d(K, L) = \sup_s |K(s) - L(s)|$

The following Lemma shows that $(\mathcal{L}, d)$ is a metric space.

Lemma 32 *d as defined above is a metric on the set L..*

The distance is always non-negative. It is also easy to check that $d(K, L) = 0$ iff $K = L$. The symmetry and triangle inequality are also easy to verify.

We further observe that $\mathcal{L}$ is a *complete* metric space in the following Lemma.

Lemma 33 *$\mathcal{L}$ is a* complete *metric space.*

Let $\{L_i\}$ be a Cauchy sequence of p-languages. We will show that this sequence converges to a p-language. As $\mathcal{L}$ is a subspace[3]of $[0, 1]^\omega$ in the metric d, and $[0, 1]^\omega$ is a complete metric space in the metric d as defined above, we conclude that the sequence converges to an element $L_\infty \in [0, 1]^\omega$. Our proof obligation is to show that L_∞ is a p-language. We now show that
for all s : $\quad \sum_a L_\infty(sa) \le L_\infty(s)$.
For any s,

[3]By closure under choice operator, we can deduce that $\mathcal{L}$ is a convex set

$\sum_a L_\infty(sa)$
$=$ { definition L_∞ monotone }
$\sum_a \lim_{i\to\infty} L_i(sa))$
$=$ { finite sum }
$\lim_{i\to\infty} \sum_a L_i(sa))$
$\leq$ { $L_i \in \mathcal{L}$ }
$\lim_{i\to\infty} L_i(s))$
$=$ {definition L_∞ }
$L_\infty(s)$.

We now show that $+_e K$ is a contraction for any K and any e.

Lemma 34 *$+_e K$ is a continuous function in the topology $(\mathcal{L}, d)$. It is a contraction for $e \neq 0$.*

Proof:
for any $X_1, X_2, K \in \mathcal{L}$,
$d(K +_e X_1, K +_e X_2)$
$=$ { definition $+_e$, definition d }
$\sup_s |eK(s) + e'X_1(s) - eK(s) - e'X_2(s)|$
$=$ { simplifying }
$e' \sup_s |X_1(s) - X_2(s)|$
$=$ { definition d }
$e'd(X_1, X_2)$
Therefore, $+_e K$ is a continuous function. Further, it is a contraction for $e \neq 0$. □

From the above theorem we obtain that any function composed of $+_e$ has a unique fixed point from Banach fixed-point Theorem.

We now show that $._e K$ is a continuous function for any K and any e.

Lemma 35 *$._e K$ is a continuous function in the topology $(\mathcal{L}, d)$. It is a contraction for $e \neq 1$.*

Proof:
We use $C(s) = (K - \Lambda(K))(s)$ in the following proof.
for any $X_1, X_2 \in \mathcal{L}$,
$d(K ._e X_1, K ._e X_2)$
$=$ { definition $._e$, definition d }
$\sup_s |K(s) - eC(s) + eC \circ X_1(s) - K(s) + eC(s) - eC \circ X_2(s)|$
$=$ { simplifying }
$e \sup_s |C \circ X_1(s) - C \circ X_2(s)|$
$=$ { definition $\circ$}
$e \sup_s | \sum_t C(t)(X_1 - X_2)(s/t)|$
$\leq$ { $\sum_u C(u) \leq 1$ and $\forall u : 0 \leq C(u) \leq 1$}
$e \sup_s | \sup_t (X_1 - X_2)(s/t)|$
$=$ { simplifying }
$e \sup_s |X_1(s) - X_2(s)|$
$=$ { definition d }
$ed(X_1, X_2)$
Therefore, $._e K$ is a continuous function. Further, it is a contraction for $e \neq 1$. Thus, $._e K$ also has a unique fixed point. □

9 Appendix B

In this section, we define some additional operators which can be used for constructing p-languages.

product
This operator captures simultaneous execution of two systems. Given two p-languages K and L, the combined system is denoted by $K \times L$ defined as
$K \times L(s) = K(s)L(s)$.
It is easy to verify that the p-language defined above satisfies constraints (C1) and (C2).

Reversal
This operator captures execution of a system in a reverse order. This operator is easier to define using completion pdf. Let C be a completion pdf for any language L, then $C^r(s)$ is defined to be same as $C(s^r)$ where s^r represents the reversal of s.

After
This operator captures the information that some string t has already been executed in the system. Let L be any p-language. The system L after executing of string t is denoted by L/t. It is defined as
$L/t(s) = L(ts)/L(t)$
We again leave it to readers to verify that the new system satisfies (C1) and (C2).