Towards Automatic Measurement of Probabilistic Processes

Lin Song, Yuxin Deng, Xiaojuan Cai
Department of Computer Science, Shanghai Jiao Tong University, China
Email: {colin_05, yuxindeng, cxj}@sjtu.edu.cn

Abstract

In this paper we propose a metric for nite processes in a probabilistic extension of CSP. The kernel of the metric corresponds to trace equivalence and most of the operators in the process algebra is shown to satisfy non-expansiveness property with respect to this metric. We also provide an algorithm to calculate the distance between two processes to a prescribed discount factor in polynomial time. The algorithm has been implemented in a tool that helps us to measure processes automatically.

Index Terms

Probabilistic processes, metric, trace equivalence, algorithm.

I. Introduction

Quantitative transition systems extend the usual transition systems, by labeling transitions with both actions and numbers. It has been argued that notions of exact behavioral equivalence such as trace equivalence do not t well with quantitative transition systems. The standard notion of trace equivalence treat the quantities as labels, this means that processes that differ from a very small probability would be considered just as different as processes that perform completely different actions. To nd a more exible way to identify processes, we want an "approximate" notion of equality of processes. Researchers have used the notion of metric to express the similarity of the behavior of processes. The smaller the distance, the more alike the behavior is. In particular, the distance between two processes is 0 if they are indistinguishable.

Much work has been reported recently in this eld. The rst proposal based on metric was made by Giacalone et al. [1] for deterministic probabilistic processes. Later, van Breugel et al. [2], [3] and Desharnais et al. [4], [5] investigated the notion of metric for more general probabilistic systems involving probabilistic distributions, nondeterminism and recursion. In [2], [3], van Breugel and Worrell de ned a metric via the terminal coalgebra of a functor and studied the ne structure of their metric whose dual problem can be reduced to a particular linear programming problem: the transshipment problem. In [4], [5], Desharnais et al. studied the metrics for labeled Markov chains and labeled concurrent Markov chains respectively and de ned the intended metric as the greatest xed point of a monotonous function. Deng et al. [6] de ned a notion of state-metric, which extended the approach of [4], [5], to a more general framework called action-labeled quantitative transition systems.

In this paper, we measure the distance of processes in a nite probabilistic process algebra, pCSP, investigated in [7], [8]. We represent the operational behavior of a process as a nite acyclic directed graph and interpret it as a set of distributions on maximal traces. We rst de ne a notion of metric for traces. Then we lift it to distributions of traces by Hutchinson metric [9]. Next, we lift it again to be a metric for sets of distributions of traces using Hausdorff distance. Finally, the distance of two processes is determined by the distance of two sets of distributions of traces that they represent. We show that the kernel of the metric corresponds to trace equivalence and all operators of the process algebra, except for parallel composition, satisfy *non-expansiveness* property with respect to this metric, i.e. the distance between two processes does not expand when they are put in the same context. We also give an algorithm to calculate the distance between two processes to a prescribed $discount\ factor$ in polynomial time. We have developed an tool that implements this algorithm and helps to measure processes automatically.

The rest of the paper is organized as follows. In Section II, we introduce pCSP and give the notion of trace equivalence between processes. In Section III, we describe our metric on processes and relate it to trace equivalence. Some non-expansiveness properties are proved in Section IV. The reduction of calculating the metric to a transshipment problem is presented in Section V. In Section VI, we present an algorithm. Some examples and comparison with other metrics are discussed in Section VII. In Section VIII, we conclude and discuss some future work.

II. Finite Probabilistic CSP

We brie y introduce a simple probabilistic process algebra based on CSP [10]; following [7], [8], we call it pCSP. We give its operational semantics in terms of a probabilistic labeled transition system and de ne a notion of trace equivalence between processes.

A. Syntax and Semantics

Let Act be a finite set of actions, ranged over by $a, b, \cdots$, which processes can perform. Then the finite probabilistic CSP processes are given by the following two-sorted syntax:

$$\begin{array}{rcl} P & ::= & S \mid P \,{}_p\!\oplus P \\ S & ::= & \mathbf{0} \mid a.P \mid P \sqcap P \mid S \,\Box\, S \mid S \,|_A\, S \end{array}$$

Here $P \,{}_p\!\oplus Q$, for $p \in (0,1]$, represents a *probabilistic choice* between P and Q: with probability p it will act like P and with probability $1-p$ it will act like Q. Any process is a probabilistic combination of state-based processes (the sub-sort S above) built by repeated application of the operator ${}_p\!\oplus$. The state-based processes have a CSP-like syntax, involving the stopped process $\mathbf{0}$, action prefixing $a._-$, *internal-* and *external choices* $\sqcap$ and $\Box$, and a *parallel composition* $|_A$ for $A \subseteq Act$.

The process $P \sqcap Q$ will first do a so-called *internal action* $\tau \notin Act$, choosing *nondeterministically* between P and Q. Therefore $\sqcap$, like $a._-$, acts as a *guard*, in the sense that it converts any process arguments into a state-based process.

The process $P \,\Box\, Q$ on the other hand does not perform actions itself, but merely allows its arguments to proceed, disabling one argument as soon as the other has done a visible action. In order for this process to start from a state rather than a probability distribution of states, we require its arguments to be state-based as well; the same applies to $|_A$. Expressions $P \,\Box\, Q$ and $P \,|_A\, Q$ for processes P and Q that are *not* state-based are therefore syntactic sugar for an expression in the above syntax obtained by distributing $\Box$ and $|_A$ over ${}_p\!\oplus$.

Finally, the expression $P \,|_A\, Q$, where $A \subseteq Act$, represents processes P and Q running in parallel. They may synchronise by performing the same action from A simultaneously; such a synchronisation results in τ. In addition P and Q may independently do any action from $(Act \backslash A) \cup \{\tau\}$.

We write pCSP for the set of process terms defined by this grammar, and sCSP for the subset comprising only the state-based process terms. The full language of CSP [10] has many more operators; we have simply chosen a representative selection, and have added probabilistic choice. Our parallel operator is not a CSP primitive, but it can easily be expressed in terms of them. It can also be expressed in terms of the parallel composition, renaming and restriction operators of CCS.

As usual the prefixing operator $a._-$ binds stronger than any binary operator; and precedence between binary operators is indicated via brackets or spacing.

The above intuitions are formalised by an *operational semantics* associating with each process term a graph-like structure representing its possible reactions to users' requests: we use a generalisation of labelled transition systems [10] that includes probabilities.

A (discrete) probability distribution over a set S is a function $\Delta : S \to [0,1]$ with $\sum_{s \in S} \Delta(s) = 1$; the *support* of Δ is given by $\lceil \Delta \rceil = \{\, s \in S \mid \Delta(s) > 0 \,\}$. We write $\mathcal{D}(S)$, ranged over by Δ, for the set of all distributions over S with finite support. We also write $\overline{s}$ to denote the point distribution assigning probability 1 to s and 0 to all others, so that $\lceil \overline{s} \rceil = \{s\}$.

We now give the probabilistic generalisation (pLTSs) of labelled transition systems (LTSs):

Definition 1: A *probabilistic labelled transition system* is a triple $\langle S, Act_\tau, \rightarrow \rangle$, where

1) S is a set of states
2) Act_τ is a set of actions Act, augmented by $\tau \notin Act$; we let a range over Act and α over Act_τ.
3) relation $\rightarrow$ is a subset of $S \times Act_\tau \times \mathcal{D}(S)$.

As with LTSs, we usually write $s \xrightarrow{\alpha} \Delta$ for $(s, \alpha, \Delta) \in \rightarrow$. An LTS may be viewed as a degenerate pLTS, one in which only point distributions are used.

We now define the operational semantics of pCSP by means of a particular pLTS $\langle \text{sCSP}, Act_\tau, \rightarrow \rangle$, constructed by taking sCSP to be the set of states and interpreting pCSP processes P as distributions $\llbracket P \rrbracket \in \mathcal{D}(\text{sCSP})$ as follows:

$$\begin{array}{rcll} \llbracket s \rrbracket & := & \overline{s} & \text{for } s \in \text{sCSP} \\ \llbracket P \,{}_p\!\oplus Q \rrbracket & := & \llbracket P \rrbracket \,{}_p\!\oplus \llbracket Q \rrbracket & \end{array}$$

Note that for each $P \in$ pCSP the distribution $\llbracket P \rrbracket$ is finite, i.e. it has finite support. The definition of the relations $\xrightarrow{\alpha}$ is given in Figure 1. These rules are very similar to the standard ones used to interpret CSP as an LTS [10], but modified so that the result of an action is a distribution.

B. PAD and NPAD graphs

We graphically depict the operational semantics of a pCSP expression P by drawing the part of the pLTS defined above that is reachable from $\llbracket P \rrbracket$ as a finite graph, which we call nondeterministic probabilistic acyclic directed (NPAD) graph. Our NPAD graphs involve two sorts of nodes, states and distributions, denoted by $\bullet$ and $\circ$ respectively. For any state s and distribution Δ with $s \xrightarrow{\alpha} \Delta$ we draw an edge from s to Δ, labeled with α. For any distribution Δ and state s in $\lceil \Delta \rceil$, the support of Δ, we draw an edge from Δ to s, labeled with $\Delta(s)$. Therefore, NPAD graphs are depicted with edges labeled by actions and probabilities alternately. In general, a state may have multiple outgoing distributions because of nondeterminism.

We now consider execution trees of an NPAD graph under particular *adversaries* which help resolving nondeterminism. The approach we adopt here is similar to the one proposed in [11] for probabilistic automata. The idea behind is that at most one

$$a{:}P \xrightarrow{a} \llbracket P \rrbracket$$

$$\frac{P \uplus Q \to \llbracket P \rrbracket}{P \uplus Q \to \llbracket Q \rrbracket}$$

$$\frac{s_1 \to}{s_1 \; s_2 \to \overline{s_2}} \qquad \frac{s_2 \to}{s_1 \; s_2 \to \overline{s_1}}$$

$$\frac{s_1 \xrightarrow{a}}{s_1 \; s_2 \xrightarrow{a}} \qquad \frac{s_2 \xrightarrow{a}}{s_1 \; s_2 \xrightarrow{a}}$$

$$\frac{s_1 \to \quad a \in A}{s_1 \mid_A s_2 \to \quad \mid_A \overline{s_2}} \qquad \frac{s_2 \to \quad a \in A}{s_1 \mid_A s_2 \to \overline{s_2} \mid_A}$$

$$\frac{s_1 \xrightarrow{a} {}_1 ; s_2 \xrightarrow{a} {}_2 \quad a \in A}{s_1 \mid_A s_2 \to {}_1 \mid_A {}_2}$$

Fig. 1
OPERATIONAL SEMANTICS OF pCSP.

action is allowed for each state. More precisely, an adversary for an NPAD graph is a function that associates to each state a distribution which is selected. An execution tree, which we call probabilistic acyclic directed (PAD) graph, is obtained from an NPAD graph by pruning all the edges corresponding to distributions which are not selected by the adversary.

Given a pCSP expression P, we use $\mathcal{G}(P)$ to denote the set of PAD graphs obtained from the NPAD graph which represents P. Consider the process

$$P = a{:}0 \;\; {}_{1=3} \; ((b{:}0 \uplus c{:}0) \;\; a{:}0)$$

its NPAD graph and some PAD graphs are illustrated in Figure 2 (a) and (b) respectively. In (a) we interpret $\llbracket a{:}0 \;\; {}_{1=3} \; ((b{:}0 \uplus c{:}0) \;\; a{:}0) \rrbracket$ to be $\llbracket a{:}0 \rrbracket$ and $\llbracket (b{:}0 \uplus c{:}0) \;\; a{:}0 \rrbracket$, which is a two-point distribution mapping the states $a{:}0$ and $(b{:}0 \uplus c{:}0) \;\; a{:}0$ to the probability 1/3 and 2/3, respectively. The end of (a) is obtained by further interpreting these two states using the rules in Figure 1. Details on how to obtain a set of PAD graphs from an NPAD graph are discussed in Section VI.

(a) An NPAD graph (b) Some PAD graphs

Fig. 2
NPAD/PAD GRAPHS

C. Trace equivalence

Given a PAD graph G, we can consider it as a set of strings $\mathcal{S}(G)$ obtained from each maximal path, that is, a set of strings of the form of $p_0 \; {}_1 \; p_1 \; {}_2 \cdots {}_n \; p_n$ where ${}_i \in \mathrm{Act}$ and $p_i \in (0, 1]$. We introduce two mappings on $\mathcal{S}(G)$.