Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 174 (2007) 95–108
www.elsevier.com/locate/entcs

Probabilistic Configuration Theories

Alessandro Tiberi[1]

Department of Computer Science
University of Rome, "La Sapienza"
Rome, Italy

Abstract

We introduce a new framework for describing computations which are both concurrent and probabilistic. This framework is a natural extension of the *Configuration Theories* of [4], and allows to express properties about both the *causal* and the *probabilistic* aspect of concurrent computations. Computations are described in an axiomatic way, by using a probabilistic extensions of poset sequents. A scheme of structural rules on these sequents is introduced and shown to be sound, while completeness can be obtained by adding a rule which is not treated in the present work. We also introduce a new probabilistic extension of *Configuration Structures*, which we use as models for probabilistic sequents.

Keywords: probabilistic semantics, concurrency, configuration structures, sequent calculus

1 Introduction

In concurrent and distributed settings, randomization plays an important role. It is folklore knowledge that there are some problems which without a source of random bits can not be solved at all, while they become almost easy when there is a source of randomness at disposal (see for instance [6,10,3].) Beside providing an important tool for solving such problems, randomness appears to be quite useful for modeling various aspects of distributed systems. In fact many features typical of distributed systems such as hardware failures, message arrivals, requests of services can be naturally described using randomized models. Thus, it is not surprising that during the last years the concurrency community has showed an increasing interest in the probabilistic aspect of computation, producing works ranging from the extension of classical process calculi (e.g. [9]), to works which adjoin probabilities to causal models of concurrent systems ([5,1,2]). We propose a novel way to axiomatize properties of concurrent and probabilistic processes, using an extension of the poset sequents of [4]. States of computation are described by sequents whose components are sequences of partially ordered sets of events. Sequents are decorated by

[1] Email: tiberi@di.uniroma1.it

1571-0661
doi:10.1016/j.entcs.2006.11.041

a matrix of monomorphisms which describes how, any computation matching the left hand side of the sequent, must extend to a computation that matches at least one poset of the right hand side. These sequents are also equipped with a vector of probabilities, which associates a probability to each poset appearing in the right hand side of the sequent. Roughly speaking, this probability is a lower bound to the probability with which any state of the computation matching the l.h.s. of the sequent will eventually evolve in a state that matches also its r.h.s. We also propose a novel probabilistic extension of *Configuration Structures* [7], which we use as *models* for the theories generated by these sequents. These structures are essentially the *monotone* structures of [4] (called in that paper *conservative*), to which we add an underlaying *finite Markov chain*, which allows us to reason about the probability with which a configuration C eventually turns into another configuration D.

In section 2 we introduce the notion of *probabilistic poset sequent*, in section 3 we formulate our extension of configuration structures, defining precisely when a structure satisfies a sequent and providing some examples, both of sequents and of probabilistic configuration structures. In section 4 the rules given in [4] are suitably modified to take into account probabilities and they are shown to be sound. In the last section we briefly discuss some possible future work.

2 Probabilistic Poset Sequents

The notion of poset sequent is introduced in [4]. Speaking informally, a poset sequent is made of two sequences of posets tied together by a matrix of monomorphisms. Intuitively, such a sequent describes some computational properties (given by the posets on the right hand side of the sequent) which must hold for any computation matching the left hand side of the sequent. We extend this notion to *probabilistic computations* by allowing the properties described by the sequents to hold with a certain probability. Before giving any formal definition and providing some more intuition about the meaning of these new sequents, we introduce some notation which essentially follows [4]. We use $\Gamma, \Delta \dots$ to denote sequences of posets, $A, B \dots$ for single posets and $a, b \dots$ for their elements. Concatenation of sequences Γ and Δ is written Γ, Δ. Matrices of monos are denoted by $\rho, \sigma \dots$. Given two matrices ρ and σ respectively of sizes $m \times n$ and $r \times n$, we write $\rho; \sigma$ for the $(m + r) \times n$ matrix that is obtained by placing ρ above σ, while if σ is of size $m \times r$ we write ρ, σ for the $m \times (n + r)$ matrix obtained by placing ρ before σ. For a formal definition of this constructions we refer the reader to [4]. Given two sequences of posets $\Gamma = A_1, \dots A_m$ and $\Delta = B_1, \dots B_n$ we write $\rho : \Gamma \to \Delta$ to denote a $m \times n$ matrix of monos such that $\rho_{ij} : A_i \to B_j$. We also use δ to denote a vector of elements $\delta_i \in [0, 1]$. Given two such vectors δ and δ', we write $\delta \cdot \delta'$ for the vector that is obtained by their concatenation. We can now define what is a probabilistic poset sequent.

Definition 2.1 A probabilistic poset sequent $\Gamma \vdash^{\delta}_{\rho} \Delta$ consists of two finite sequences of posets Γ and Δ, a matrix of monos $\rho : \Gamma \to \Delta$ and a vector δ such that if $\Delta = \Delta_1, \dots, \Delta_n$ then $\delta = \delta_1 \dots \delta_n$ and $\delta_i \in [0, 1]$.

The reader will probably better understand this definition after the notion of interpretation and the model on which we interpret these sequents are given. However, before proceeding further, we find it useful to build some intuition about the meaning of these sequents. Given a sequent $\Gamma \vdash_\rho^\delta \Delta$, its left hand side Γ describes, in terms of *causal* relations between events a computational status. Events are elements of the posets in Γ, and their causal relations are represented by partial orders. Each Γ_i should be thought of as a property of the computational status described by Γ. Thus, each Γ_i is somehow in conjunction with the others. The sequence Δ instead describes properties that must eventually be satisfied by any computation C for which all the "properties" Γ_i holds. In particular we require that C will reach a status D that satisfies at least *one* Δ_i (thus each Δ_i should be thought of as being in disjunction with the others). We also require that the probability of reaching D from C is at least δ_i. This last constraint is essentially what makes the difference between a poset sequent and a probabilistic poset sequent. Concrete examples of sequents will be given in section 3.

3 Probabilistic Configuration Structures

In this section we introduce the model on which probabilistic sequents are interpreted.

We start by recalling the definition of *configuration structure* [7]:

Definition 3.1 A configuration structure is a pair $(E, \mathcal{C})$ where E is a set, whose elements are called *events*, and $\mathcal{C}$ is a family of subsets of E, called *configurations*.

Given a configuration structure $(E, \mathcal{C})$ and a configuration $C \in \mathcal{C}$, we denote with $Sub(C)$ the set of configurations of $\mathcal{C}$ which are contained in C. A partial order $\leq_C$ can be associated in a natural way to each configuration by defining $\leq_C = \{(a, b) | \forall D \in Sub(C), b \in D \implies a \in D\}$. In [4] a special class of configuration structures, called *monotone*, is introduced. These structure are required to enjoy the following properties:

- *finiteness*: if an event belongs to a configuration in C, then it also belongs to a finite subconfiguration of C.
- *coincidence-freeness*: if two distinct events a, b belong to a configuration C, then there exists $D \in Sub(C)$ containing exactly one event in $\{a, b\}$.
- *non-emptiness* of $\mathcal{C}$
- *downwards-closed* bounded intersection: $\forall C \in \mathcal{C}$, $\forall D, F \in Sub(C)$ and for all $a \in D \cap F$ if $b \leq_D a$ then $b \in F$.

As shown in [4] monotone structures preserve the partial orders induced by subconfiguration, that is if $D \in Sub(C)$ and $a \leq_D b$ then $a \leq_C b$ and hence for these structures inclusion can be seen as a monomorphism between posets. In the rest of the paper we assume all structures to be monotone and we also require *connectedness*: for all non-empty $C \in \mathcal{C}$ there exists $a \in C$ such that $C \setminus \{a\} \in \mathcal{C}$. Connectedness implies *rootedness*: the empty configuration belongs to $\mathcal{C}$. If we also

require closure under *bounded union* (that is if $C, D \subseteq D'$ then $C \cup D$ is a configuration), then if we consider a configuration C and one of its subconfiguration D such that $C \setminus D = \{e_1, \ldots, e_n\}$, there is at least one sequence of events $e_{\pi(1)} \ldots e_{\pi(n)}$ such that $D \cup \bigcup_{i \leq k} \{e_{\pi(i)}\}$ is a configuration for all $k \leq n$ and π is a permutation on $[n]$. This last requirement is not strictly necessary for our theory, but it makes some definitions less technical, hence we enforce it. However, we underline that it is not essential.

Given a configuration structure $\mathcal{C}$, we call an event e *enabled* at a configuration $C \in \mathcal{C}$ if $e \notin C$ and $\exists D \in \mathcal{C}$ s.t. $D = C \cup \{e\}$. We denote with $E(C)$ the set containing all the events enabled at C. Informally speaking, a *Probabilistic Configuration Structure* is a *Configuration Structure* enriched with a few components which allows to reason about the probability of reaching a configuration C starting from any of its subconfigurations. First we attach to all events a *label*, and we assume the set $\mathcal{L}$ of labels to be finite. Next we attach to each configuration a probability function on E. Finally we associate to this structure a *finite Markov Chain* (see for instance [8]), which we see as a directed weighted graph whose edges are labeled with elements of $\mathcal{L}$. Our structure is then mapped on this graph, so that the image of each configuration is a node and if $S(C)$ is the node on which a configuration is mapped, the set $E(C)$ is mapped on the outgoing edges of $S(C)$. We require this mapping to preserve labels, probabilities and "transitions". In other words we enforce this mapping to be a kind of homomorphism between the structure seen as a (possibly infinite) graph whose nodes are configurations and edges enabled events, and the graph representing the associated Markov Chain. This is made precise in the following definitions. We first define what is a *finite labeled Markov chain*. In the following we fix the set of labels to be $\mathcal{L}$.

Definition 3.2 A finite labeled Markov chain $\mathcal{M}$ is a directed graph $G = (S, A)$ together with a weight function $w : A \to [0, 1]$ and a labeling function $L_M : A \to \mathcal{L}$. An edge from a node s_i to a node s_j is denoted by s_{ij}, and the weight function is subject to the constraint that for all $s_i \in S$,

$$\sum_{s_{ij} \in A} w(s_{ij}) = 1.$$

We call the elements of S *states*.

Definition 3.3 A probabilistic configuration structure is a 6-tuple

$$(E, \mathcal{C}, \mathcal{P}, L_C, \mathcal{M}, h)$$

where

(i) $(E, \mathcal{C})$ is a configuration structure

(ii) $\mathcal{P} : \mathcal{C} \to (E \to [0, 1])$ is a function which assigns to each configuration C a function P_C such that $\sum_{e \in E(C)} P_C(e) = 1$

(iii) $L_C : E \to \mathcal{L}$ is a labeling function

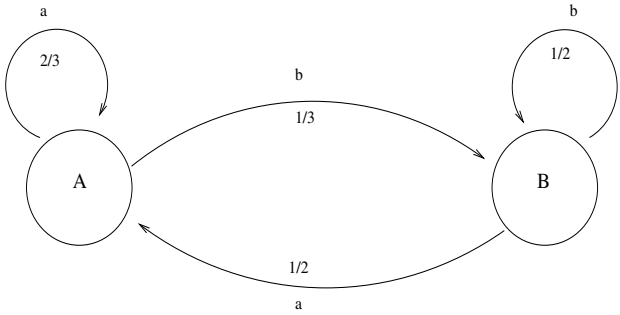

Figure 1. A Markov chain associated to the structure of example 3.4. A configuration whose last event is labeled with an a is mapped on state A, all the others on state B.

(iv) $\mathcal{M}$ is a finite labeled Markov chain

(v) h is a function that maps configurations of $\mathcal{C}$ on states of $\mathcal{M}$ and the disjoint union of the events enabled at each configuration on arcs of M. We write $h_C(e)$ to mean $h(e)$ where $e \in E(C)$. We also require that if $h(C) = s_i$ then $\forall\, e \in E(C)$ s.t. $P_C(e) > 0$ there exists an arc s_{ij} s.t. :
- $h_C(e) = s_{ij}$
- $w(s_{ij}) = P_C(e)$
- $L_M(s_{ij}) = L_C(e)$
- $h(C \cup \{e\}) = s_j$

Example 3.4 Consider the structures whose events is the union of the two countably infinite set $A = \{a_1, a_2, \dots\}$ and $B = \{b_1, b_2, \dots\}$, where events in A are labeled with a and events in B with b. Let the configurations of these structure be all the sets C (including the empty set) s.t. if $|C| = n$ then $C = \{c_1, c_2, \dots, c_n\}$ where $c_i \in \{a_i, b_i\}$. Moreover, for such a configuration let the set of enabled events to which a positive probability is attached, be $\{a_{n+1}, b_{n+1}\}$ and let $P_C(a_{n+1}) = 2/3$ and $P_C(b_{n+1}) = 1/3$ if the last event in C is $c_n = a_n$ while $P_C(a_{n+1}) = P_C(b_{n+1}) = 1/2$ otherwise. By mapping this structure in the obvious way on the Markov chain depicted in figure 1 we obtain a probabilistic configuration structure.

Once we have mapped a configuration structure on a Markov chain, we can define the probability of reaching a configuration D, starting from one of its subconfiguration C. We recall that a transition matrix $\mathbf{P}$ can be associated to any finite Markov chain. An entry p_{ij} of this matrix is the probability of reaching state s_j from state s_i (the weight of the edge s_{ij}) in one step. We also recall that the probability of reaching state s_j starting from state s_i in n steps is defined as the jth entry of the vector $\mathbf{u}^{(n)}$, where

$$\mathbf{u}^{(n)} = \mathbf{u}\mathbf{P}^n$$

$\mathbf{u}$ is a vector with all zero components except for the ith entry which is equal to 1 and $\mathbf{P}^n$ is the $\mathbf{P}$ raised to the n.

Definition 3.5 Let $\mathcal{C}_p = (E, \mathcal{C}, \mathcal{P}, L_C, \mathcal{M}, h)$ be a probabilistic configuration structure, and let C, D be two configurations such that $C \in Sub(D)$ and $|D \setminus C| = n$. If

$h(C) = s_i$ and $h(D) = s_j$, then the probability of reaching a configuration that is mapped on the same state s_j of D starting from C is defined as

$$Pr(C \to h(D)) = \mathbf{u}_j^{(n)}$$

where $\mathbf{u}^{(n)} = \mathbf{u}\mathbf{P}^n$, $\mathbf{P}$ is the transition matrix of $\mathcal{M}$ and $\mathbf{u}$ is zero everywhere except for its ith entry which is equal to one.

Notice that by the above definition the probability $Pr(C \to C)$ is always equal to 1.

This definition considers somehow equivalent configurations mapped on the same state of the underlaying Markov chain. Actually, for the notion of satisfaction that we have in mind this is too abstract, since we want to compute precisely the probability of going from a configuration C to a configuration D. To this aim, we need to introduce few other definitions.

Definition 3.6 A path $\pi_{C,D}$ from a configuration C to a configuration D is a sequence of pairwise distinct events $\pi = e_1 e_2 \cdots e_n$ such that $e_i \notin C$, $C \cup \bigcup_{i \le k} e_i = D_k \in \mathcal{C}$ for $k \in [n]$ and $D_n = D$. C is called the source of π and D the destination.

The set of all paths from C to D is written $\Pi_{C,D}$. We write $\pi_{c,*}$ to denote a path whose source is C and destination is not specified. We also write $|\pi|$ to denote the length of π. We can then define the probability of a path $\pi_{C,D} = e_1 \dots e_k$ as

$$Pr(\pi_{C,D}) = w(h_C(e_1)) \cdot Pr(\pi_{C \cup \{e_1\},D})$$

where $\pi_{C\cup\{e_1\},D} = e_2 \dots e_k$.

We call *dead* a path $\pi_{C,D}$ if D is a configuration with no events enabled (and hence in the Markov chain it is represented by a node with only one ingoing edge of weight 1). We can extend a dead path $\pi_{C,D} = e_1 \dots e_k$ of length k, to a path $\pi'_{C,D}$ of length $k + c$ by allowing D to perform c *invisible* events ϵ with probability one. Thus for all configurations D with no events enabled, if $h(D) = s_i$ we extend h_D so that $h_D(\epsilon) = s_{ii}$ and $P_D(\epsilon) = 1$.

A Probabilistic Configuration Structure entails a probability function on the set of paths of any fixed length n which share the same source C, but only after we extend the dead paths $\pi_{C,D}$ whose length is less than n to paths $\pi'_{C,D}$ of length n by adding a suitable sequence of invisible events ϵ.

Remark 3.7 Let the set $\Pi_{C,k}$ be defined as the set containing all paths of length k, whose source is C and all dead paths $\pi_{C,*}$ whose length is less than k extended to path of length k. It holds that:

(i) $\forall \pi \in \Pi_{C,k}\, Pr(\pi) \ge 0$

(ii) $\forall \pi \in \Pi_{C,k}\, Pr(\pi) \le 1$

(iii) $Pr[\Pi_{C,k}] = \sum_{\pi \in \Pi_{C,k}} Pr(\pi) = 1$

We omit the proof since it is routine. The reader should notice that the probability defined in 3.5, $Pr(C \to h(D))$, is an upper bound to the probability $Pr[\Pi_{C,D}]$.

This property can be useful for proving that a configuration structure *does not* satisfy a sequent by looking only at the Markov Chain, without considering paths.

We can now define when a probabilistic configuration structure satisfies a probabilistic poset sequent. We call a $n \times 1$ matrix of monos $\pi : \Gamma \to C$ an *interpretation* of Γ in C. The following definition is similar to the one given in [4] but it also takes into account probabilities.

Definition 3.8 A probabilistic configuration structure $(E, \mathcal{C}, \mathcal{P}, L_C, \mathcal{M}, h)$ is said to satisfy a probabilistic sequent $\Gamma \vdash^{\delta}_{\rho} \Delta$ when, for any $C \in \mathcal{C}$ and interpretation $\pi : \Gamma \to C$, there exists $D \in \mathcal{C}$, a component Δ_k and a mono $q : \Delta_k \to D$ such that $C \in Sub(D)$ and, for all i, the following diagram commutes and $Pr[\Pi_{C,D}] \geq \delta_k$

$$\begin{array}{ccc} \Gamma_i & \xrightarrow{\rho_{ik}} & \Delta_k \\ {\scriptstyle \pi_i}\downarrow & & \downarrow{\scriptstyle q} \\ C & \xrightarrow{\subseteq} & D \end{array}$$

Note that since we assume our structure to be monotone, the inclusion $\xrightarrow{\subseteq}$ of the above diagram is a monomorphism between posets. Both the notion of sequent and of satisfaction can be extend to a setting in which events are labeled, just by enforcing the additional constraint that all monos preserve labels. In the following we will use labeled sequents. In the remaining of this section we develop a simple example to give the reader a taste of what these sequents can be used for.

3.1 Example

Consider a scenario in which there is a shared resource on which processes can make some computation, but only after they have acquired a lock on it. Sequents can be used to axiomatize the correctness of computation (e.g. exclusive access to the shared resource) and, as we shall see, also to enforce processes to be somewhat fair: in fact by using suitable sequents we can prescribe processes to release the resource with a *certain probability* after they have made some computation on the resource itself. For the following sequents three labels are used, $\mathcal{L} = \{l, u, c\}$. They stand for: *lock*, *unlock*, *compute*.

The first desirable property is that before a process can make any computation on a resource, it should acquire a lock on it. Since this is a property that essentially is about the *past* of a computation, we require it to hold with probability 1. The following sequent describes this requirement:

We also want to enforce the lock to be exclusive. Unfortunately, poset sequents are not very good at expressing such a property, but if we restrict ourself to a setting in which no configuration will be reached with probability one, the following sequent does the work:

This sequent expresses the required property as long as we forbid any configuration to evolve in some other configuration with probability 1, since in this case this sequent enforces the unlock action to happen *before* one of the two locks.

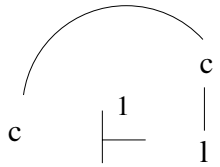

Figure 2. This sequent is made of two posets: the one of the l.h.s contains a single element labeled with c, the other contains two elements, b, d, labeled respectively with l and c, and $b < d$. The matrix ρ is made of a single mono, which maps a on d (depicted as the arc in the figure)

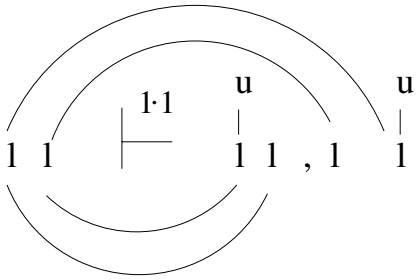

Figure 3. This sequent contains three posets, one in the l.h.s and two (separated by a comma) in the r.h.s., this time ρ is a vector of two monos, one depicted in the upper part of the sequent and one in the lower. This sequent says that whenever the resource has been locked twice, one of the locks must have been released.

These two properties essentially make little use of probabilities. We now describe two other properties in which probability plays an important role. We want to enforce that when a process acquires a lock, then there is a positive probability (equal to 1/10) that it *does* some computation on the resource it has locked:

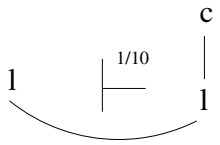

Figure 4. This sequent is similar to the first, but here we have put a probability equal to 1/10, thus any computation in which a lock has been acquired, will *eventually* extend to a computation in which the resource is used, with probability at least 1/10.

The last property we require is that once a process has acquired a lock, it does not hold it forever. In particular we ask that once it has used the resource at least once, there is a positive probability (namely 1/3) that it releases its lock. This is expressed by the following sequent:

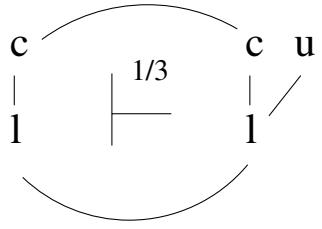

Figure 5. As for the previous one, this sequent contains two posets and a matrix of just a single mono. In the poset of the right hand side there is an unlock action causally dependent on the lock that has enabled the computation.

Now we describe a probabilistic configuration structure whose configurations satisfy all these sequents. This structure is meant to describe a concurrent computation in which two processes share a resource. Let's start with describing the events of this structure. We assume to have an infinite set of events E, which is partitioned in seven sets, L_i, C_i, U_i with $i = 1, 2$ and a set $I = \{w_1, w_2 \cdots\}$ of idle events. The set $L_1 = \{l_1^1, l_2^1 \dots\}$ contains the lock event of the first process, L_2 of the second and similarly for the other sets. We fix the set of labels $\mathcal{L} = \{l, u, c, w\}$ and associate label l to all events in the sets L^i and similarly for the other sets. To each lock action l_k^i we associate a set of computations $C_k^i \subset C^i$ whose elements we denote with $c_{k,j}^i$. For describing the configurations of our structure, we find it convenient to introduce a partial order on E: we assume that $l_k^i < c_{k,j}^i$, $l_k^i < u_k^i < l_{k+1}^i$ and $c_{k,j}^i < c_{k,j+1}^i$ for all k and $i \in \{1, 2\}$ (notice that since we assume $<$ to be a partial order, it must be transitive.) A configuration C (containing only events in E) is in our structure if and only if the relation $\leq_C$ agrees with the partial order on E and $0 \leq |C \cap (L^1 \cup L^2)| - |C \cap (U^1 \cup U^2)| \leq 1$. This last condition guarantees that in each configuration there is at most one lock action l_k^i that is not matched by the corresponding unlock action. The set of configurations defined in this way can be partitioned in three sets, one in which all configurations contain the same number of lock and unlock events (call it C_b), one whose configurations contain an event l_k^i not matched by the corresponding unlock and do
not contain the event $c_{k,1}^i$ [2] (call it C_l), and one set containing all the other configurations (call it C_c). We also enrich the configurations in C_b by allowing idle events to be sequentially (we also require $w_i < w_{i+1}$) added to them but we still keep the implicit constraint that the resulting structure is monotone and connected.

Now we have to define the probability function associated to each configuration. Consider a configuration $C \in C_b$, we assign a positive probability to a subset of its enabled event: $\{l_{k+1}^1, l_{j+1}^2, w_{i+1}\}$, where k, j, i are the greatest indices of events in L^1,L^2 and I contained in C. We associate the following probabilities to this events: $P_C(w_{i+1}) = 0.1$, $P_C(l_{k+1}^1) = P_C(l_{k+1}^2) = 0.45$. In a configuration C in C_l there are several enabled events, however, if l_k^i is the unique lock event that is not matched by the corresponding unlock, we attach a positive probability only to the events u_k^i and $c_{k,1}^i$, in particular $P_C(u_k^i) = P_C(c_{k,1}^i) = 0.5$. Similarly, if C is a configuration in C_c, l_k^i is as before and $c_{k,j}^i$ is such that no events $c_{k,h}^i$ with $h > j$ is in C, we define $P_C(u_k^i) = 0.8$ and $P_C(c_{k,j+1}^i) = 0.2$.

We can now associate to this structure the Markov chain in figure 6.

How configurations are mapped on the states of this chain should be understandable by looking at the labels given to the states: configurations in C_b are associated to the state with the same label, while configurations in C_l are associated either to state C_{l1} or C_{l2}, depending on which process has performed the "last" lock action. Configurations in C_c are mapped on the remaining state. Let's see now why this probabilistic structure satisfies, for instance, the sequent in figure 4. Consider a configuration C for which there exists an interpretation mapping the left hand side

[2] hence neither any $c_{k,j}^i$

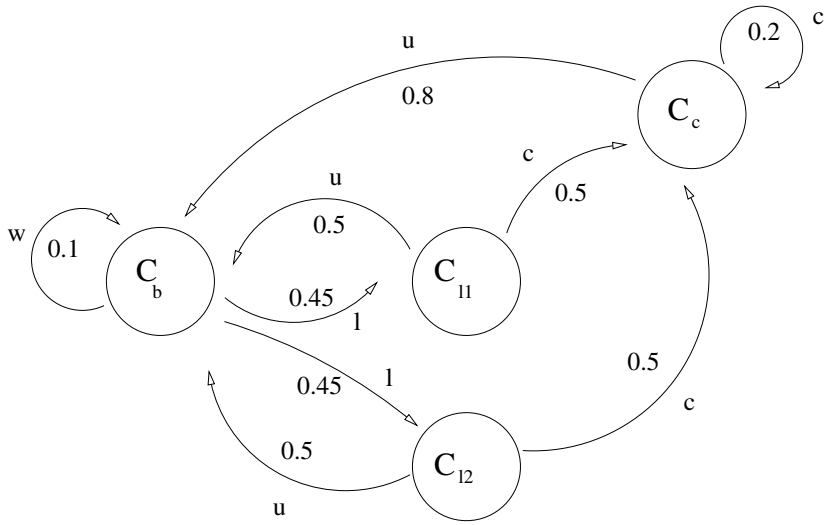

Figure 6. A Markov chain associated with the structure of the example. Edges are labeled with the name of the events and with the associated probability.

$$[\text{TRUE}]\ \frac{}{\vdash^{\delta}\ \emptyset} \qquad [\text{ISO}]\ \frac{}{A \vdash^{\delta}_{\phi}\ B}\ (\phi \text{ is iso})$$

$$[\text{L-WEAK}]\ \frac{\Gamma \vdash^{\delta}_{\rho}\ \Delta}{\Gamma, \emptyset \vdash^{\delta}_{\rho;\emptyset}\ \Delta} \qquad [\text{R-WEAK}]\ \frac{\Gamma \vdash^{\delta}_{\rho}\ \Delta}{\Gamma \vdash^{\delta\cdot\delta_A}_{\rho;\tau}\ \Delta, A}$$

$$[\text{R-CONTR}]\ \frac{\Gamma \vdash^{\delta\cdot\delta'_A\cdot\delta''_A}_{\rho,\tau,\tau}\ \Delta, A, A}{\Gamma \vdash^{\delta\cdot\delta_A}_{\rho,\tau}\ \Delta, A}\ (\delta_A = min\{\delta'_A, \delta''_A\}) \qquad [\text{L-CONTR}]\ \frac{\Gamma, A, A \vdash^{\delta}_{\rho;\tau;\tau}\ \Delta}{\Gamma, A \vdash^{\delta}_{\rho;\tau}\ \Delta}$$

$$[\text{L-EXC}]\ \frac{\Gamma, A, B, \Pi \vdash^{\delta}_{\rho;\tau;\phi;\theta}\ \Delta}{\Gamma, B, A, \Pi \vdash^{\delta}_{\rho;\phi;\tau;\theta}\ \Delta} \qquad [\text{R-EXC}]\ \frac{\Gamma \vdash^{\delta\cdot\delta_A\cdot\delta_B\cdot\delta_\Pi}_{\rho,\tau,\phi,\theta}\ \Delta, A, B, \Pi}{\Gamma \vdash^{\delta\cdot\delta_B\cdot\delta_A\cdot\delta_\Pi}_{\rho,\phi,\tau,\theta}\ \Delta, B, A, \Pi}$$

Table 1
Structural Rules I

of the sequent on C. Such a configuration must contain at least one lock event l^i_k on which the l event in the l.h.s of the sequent is mapped. We'll show that regardless of what is the state $h(C)$, there is a path of probability greater than 1/10 that C evolves to a configuration D (which makes the diagram of definition 3.8 commute). Suppose that $h(C) = C_b$, then with probability 0.45 we make an l action by means of an event l^i_{k+c}, thus moving in C_{li}. From C_{li} we can make a c action with probability 0.5 by means of an event $c^i_{k+c,1}$. Since $l^i_k < l^i_{k+c} < c^i_{k+c,1}$ the diagram commutes by mapping the c event of the r.h.s. of the sequent on $c^i_{k+c,1}$, moreover the probability of the described path is equal to 9/40 which is greater than 1/10. If instead $h(C) = C_{lj}$, we have two cases: either $j = i$ and then we can perform a c action we probability 0.6 and we are done, or we may have to go back to a configuration in C_b, by performing an unlock action and then repeat the path described before. Again the probability of this path (9/80) is greater than the required one. We skip

the case in which $h(C) = C_c$ since it is very similar to the previous one.

4 Probabilistic Configuration Theories

In [4] the notion of *Configuration Theories* is introduced. In that work a configuration theory is defined as a set of sequents closed under a certain rule scheme. Moreover, that rule scheme is shown to be *sound* and, by adding a rule and constraining the sequents to be made of finite posets, also *completeness* is proven. Here we show that the same rule scheme can be naturally adapted for probabilistic poset sequents, preserving both soundness and completeness. Before explaining the rule scheme, we need to introduce some notation. Given two matrices $\rho : \Gamma \to A$ and $\tau : A \to \Pi$ we write with $\rho\tau$ the matrix $\sigma : \Gamma \to \Pi$ whose entries are $\sigma_{ij} = \rho_{1i} \cdot \tau_{j1}$ where here $\cdot$ stands for function composition (written in diagrammatical order). Moreover, given a probabilistic poset sequent $\Gamma \vdash^{\delta}_{\rho} \Delta$, we denote the ith element of the vector δ with δ_i, while given two vectors δ, δ' with $\delta_i\delta'_j$ we denote the product of the ith element of δ with the jth element of δ', while with $\delta \cdot \delta'$ we denote the vector obtained by concatenating δ with δ'.

The rules are shown in table 1 and table 2. Following [4], we can define a *Probabilistic Configuration Theory*:

Definition 4.1 A probabilistic configuration theory is a set of probabilistic poset sequents, closed under the rule of table 1 and table 2.

We first give a intuitively explanation of the meaning of some rules, then in the next subsection we formally prove the soundness. Rule [iso], says that whenever a sequent is made of two posets which are isomorphic then this sequent is satisfied by any probabilistic configuration structure, regardless of the vector of probabilities this sequent is decorated with. This is because if a configuration interprets poset A, through an interpretation π, then by choosing the morphism q of definition 3.8 as $q = \phi^{-1}\pi$ and $D = C$, we have that the diagram commutes and the probability that C remains in C by an empty path is of course 1. Rule [r-weak] says that whenever a configuration satisfies $\Gamma \vdash^{\delta}_{\rho} \Delta$, it also satisfies the sequent obtained by adding to the r.h.s of the former a poset A, regardless of how Γ is mapped on A by τ and of the value of δ_A. The correctness of this rule easily follows from definition 3.8.

4.1 Soundness

We now show that the rules given in table 1 and table 2, are sound when interpreted in probabilistic configuration structures. That is every configuration structure that satisfies the premises of a rule, also satisfies its consequence.

Theorem 4.2 *The rule of table 1 and table 2 when interpreted on probabilistic configuration structures are sound.*

Proof We need to prove that for each rule, if a configuration structure satisfies its premises then it also satisfies its consequence. For most of the rule in table 1 the

$$[\text{L-CUT}]\ \frac{\Pi \vdash^{\delta_A}_{\tau} A \qquad \Gamma, A \vdash^{\delta}_{\rho;\phi} \Delta}{\Gamma, \Pi \vdash^{\delta'}_{\rho;\tau\phi} \Delta}\ (\text{where } \delta' = \delta_A\delta_1 \cdot \delta_A\delta_2 \cdots \delta_A\delta_n)$$

$$[\text{R-CUT}]\ \frac{\Gamma \vdash^{\delta\cdot\delta_A}_{\rho,\phi} \Delta, A \qquad A \vdash^{\delta^\Pi}_{\tau} \Pi}{\Gamma \vdash^{\delta\cdot\delta'}_{\rho,\phi\tau} \Delta, \Pi}\ (\text{where } \delta' = \delta_A\delta^\Pi_1 \cdot \delta_A\delta^\Pi_2 \cdots \delta_A\delta^\Pi_n)$$

Table 2
Structural Rules II

argument is similar to the one informally carried out for the rule [iso], hence we skip the proof for these rules.

Let us consider rule [l-cut]. Consider a configuration C that satisfies both $\Pi \vdash^{\delta_A}_{\tau} A$ and $\Gamma, A \vdash^{\delta}_{\rho;\phi} \Delta$. Let $v : \Gamma \to C$ and $\pi : \Pi \to C$, $v_A : A \to C$ be arbitrary interpretations. Notice that any interpretation $\psi : \Gamma, \Pi \to C$ can be obtained as v, π by a suitable choice of π and v. Since C satisfies the first sequent, there must be a configuration D, a mono $r = C \xrightarrow{\subseteq} D$ and a mono $q : A \to D$ such that the diagram of definition 3.8 commutes, that is for all i, $\pi_i r = \tau q$. Note that if v is an interpretation in C and $C \subseteq D$, then v is also an interpretation in D. Thus, fixed v, there must be a configuration D', a mono $r' = D \xrightarrow{\subseteq} D'$ and a mono $q' : \Delta_k \to D$, such that the corresponding diagram commutes for all components of the l.h.s. of the poset, that is $v_i r' = \rho_{ik} q'$ and $v_A r' = \phi_k q'$. Pasting together these two diagrams, choosing $v_A = q$ and noticing $(\phi\tau)_{ik} = \phi_i \tau_k$, shows that for any interpretation $\pi' : \Gamma, \Pi \to C$ we can build the required commuting diagram. It remains to show that $Pr[\Pi_{C,D'}] \geq \delta_A\delta_k$: we know that $Pr[\Pi_{C,D}] \geq \delta_A$ and $Pr[\Pi_{D,D'}] \geq \delta_k$. The thesis follows by observing that $Pr[\Pi_{C,D'}] = \sum_{\pi \in \Pi_{C,D}} Pr(\pi) \Pr[\Pi_{D,D'}]$. Consider now rule [r-cut], and assume that its premises $\Gamma \vdash^{\delta\cdot\delta_A}_{\rho,\phi} \Delta, A$ and $A \vdash^{\delta^\Pi}_{\tau} \Pi$, are satisfied by a configuration C. Thus from the satisfaction of the first sequent we know that there exists a configuration D containing C, and a mapping q which is either $q : \Delta_j \to D$ for some component Δ_j of Δ, or $q : A \to D$ which makes the required diagram commute. In the first case the thesis follows immediately. For the other case, notice that since C satisfies the second sequent, then also D must satisfy it, thus there is a configuration D' containing D and a mono $q' : \Pi_k \to D'$ such that the diagram commutes even if we take as interpretation q. Pasting together the two diagrams, shows that we are able to build a commuting diagram for the consequence of the rule. For the vector of probabilities that is associated to the consequence, exactly the same argument carried on for [l-cut] can be used. □

So far we have proven only the soundness of our rule scheme. Actually, this rule scheme is not complete: there are valid sequents which can not be derived. In [4] a new rule, called *extend*, is introduced to obtain the completeness at the cost of constraining sequents to be finite. It is not hard to import that rule in our setting and to show that in this way we obtain a complete calculus. However the rule itself is quite technical and needs the introduction of some machinery which is out of the scope of this work. We refer the interested reader to [4], being confident that the details of how the rule *extend* must be modified can be easily worked out.

5 Conclusions

We have introduced a new kind of sequents which are able to capture both the *causal* and the *probabilistic* aspects of concurrent computation, and we have interpreted these sequents in a new framework which suits these sequents in a quite natural way. We have also provided a scheme of rules which we have proven to be sound, and which can be easily extended to a be complete. However, this work is still at an initial stage, and we think that there are quite a number of things which need further investigation. Concerning our sequents, we think that the way we have adjoined probabilities is just one among the possible ones. We think that it would be interesting to explore a notion of satisfaction which takes into account the fact that in a probabilistic setting not all configurations are equal. In other words, a probabilistic configuration structure may fail satisfying a theory because of just a tiny portion of its configurations, which in a probabilistic scenario would make sense to ignore. Hence we could modify the notion of satisfaction requiring only that a "large set" (in a probabilistic sense) of the configurations satisfies the sequents. Moreover, interpreting our sequents on our probabilistic structure has not been hard, but these structures are particularly simple. We think we should explore the feasibility of using as models the probabilistic event structures of [5] which have both a causal and a probabilistic flavor. Another promising model into which we could interpret our sequent are the *probabilistic safe Petri nets* of [2] which shares several features with *Markov chains*, and thus should be not too hard to describe using our sequents.

Concerning probabilistic configuration structures, we are aware that while on one side they are quite simple and easy to handle, on the other side, they have one serious limitation: all choices are probabilistic. As noted in [11], this is probably a too strong assumption since many transition are better modeled using *nondeterminism*. Thus we think we should explore different settings in which probabilistic choice is combined with nondeterministic choice, maybe in a similar way to [11] where *concurrent Markov chains* are introduced, using a *scheduler* to model nondeterminitics steps.

References

[1] Abbes and Benveniste. Branching cells as local states for event structures and nets: Probabilistic applications. In *FOSSACS: International Conference on Foundations of Software Science and Computation Structures*. LNCS, 2005.

[2] Samy Abbes. The (true) concurrent markov property and some applications to markov nets. In Gianfranco Ciardo and Philippe Darondeau, editors, *ICATPN*, volume 3536 of *Lecture Notes in Computer Science*, pages 70–89. Springer, 2005.

[3] M. Ben-Or. Another advantage of free choice: completely asynchronous agreement protocols. In *PODC'83*, pages 27–30, 1983.

[4] Pietro Cenciarelli. Configuration theories. In *CSL '02: Proceedings of the 16th International Workshop and 11th Annual Conference of the EACSL on Computer Science Logic*, pages 200–215, London, UK, 2002. Springer-Verlag.

[5] Daniele Varacca. Probabilistic event structures and domains, July 08 2003.

[6] Fischer, Lynch, and Paterson. Impossibility of distributed consensus with one faulty process. *JACM: Journal of the ACM*, 32, 1985.

[7] R. Van Glabbeek and Gordon Plotkin. Configuration structures. In *Tenth Annual Symposium on Logic in Computer Science*, pages 199–209, San Diego, California, 1995. IEEE Computer Society.

[8] Charles M. Grinstead and J. Laurie Snell. *Introduction to Probability: Second Revised Edition*. American Mathematical Society, 1997.

[9] Oltea Mihaela Herescu and Catuscia Palamidessi. Probabilistic asynchronous pi-calculus. In *Foundations of Software Science and Computation Structure*, pages 146–160, 2000.

[10] D. Lehman and M. O. Rabin. On the advantage of free choice: A fully symmetric and fully distributed solution to the dining philosophers problem. In *Proceedings of 10th ACM Symposium of Principles of Programming Languages*, pages 133–138, Williamsburg, 1981.

[11] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS'85*, pages 327–338, 1985.