Probabilistic models for concurrency

Notes for a minicourse

Daniele Varacca
ENS-Paris

IT University Technical Report Series **TR-2005-55**

ISSN 1600–6100 **January 2005**

ISSN 1600–6100

ISBN 87-7949-079-4

Copies may be obtained by contacting:

IT University of Copenhagen
Glentevej 67
DK-2400 Copenhagen NV
Denmark

Telephone:	+45 38 16 88 88
Telefax:	+45 38 16 88 99
Web	`www.itu.dk`

Probabilistic models for concurrency
Notes for a minicourse

Daniele Varacca
ENS - Paris

Contents

What these notes are

These notes were written to accompany a minicourse given in November 2004 at the IT University of Copenhagen, in the context of the FIRST PhD school. They can be used as a quick introduction to the topic by a reader knowledgeable in concurrency theory.

What these notes are not

These notes are not meant to be complete in any way, and they are indeed quite sketchy. I wanted to stay within a reasonable size, and many issues are not even mentioned. An interested reader should follow the bibliographic links. The reader should also be warned that the names and the notation I have chosen are not always universally accepted.
If you find any mistakes, or have any suggestions for improvement, please send me a mail at `varacca@brics.dk`.

Acknowledgments

I want to thank Thomas Hildebrandt for inviting me to give this minicourse. Mikkel Bundgaard and Bartek Klin helped in removing several mistakes from these notes.

1 Introduction and motivations

Mathematics seems complicated, but the real world is much more. That is why we study mathematical models rather than the real world. Different models of computation focus on different aspects of computation: the aspects that interest us. Some models focus on sequential computation, some on security properties, some on algorithmic issues, and so on. In this course we focus on two aspects of the notion of computation: *probability* and *concurrency*.

A model of computation is probabilistic when it is able to represent different choices and to provide information on the probability of such choices. It differs from *deterministic* models, where no choice is represented, and from *nondeterministic* models, where different choices are represented, but no information is provided on how such choices are resolved.

Probabilistic choice can be considered as a refined version of nondeterministic choice. When different behaviours are available, we might decide to make a choice randomly in order to mislead some kind of "adversary". This adversary could be the one that finds the worst case for an algorithm, could be an eavesdropper, or could be a scheduling policy that maintains an undesirable symmetry. Alternatively, we might know some information on how the choice is made by the environment. In both cases we can tell with which probability each possible action is performed.

There are several algorithms which take advantage of random choices during the computation (e.g.[Rab80]). Cryptographic protocols use random choices to increase security [GM84]. In a distributed setting, we can use random choices to break symmetries [Lyn96, HP00]. On the other hand probabilistic models allow us to consider phenomena (noise, malfunction, intrusion) which in the real world can affect computations. (Probability theory is also a fundamental ingredient in the theory of quantum computation [NC00].)

A model of computation is a model for concurrency when it is able to represent systems as composed of independent autonomous components, possibly communicating with each other. The notion of concurrency should not be confused with the notion of *parallelism*. Parallel computations usually involve a central control which distributes the work among several processors. In concurrency we stress the independence of the components, and the fact that they communicate with each other. Parallelism is like ancient Egypt, where the Pharaoh decides and the slaves work. Concurrency is like modern Italy, where everybody does what they want, and all use mobile phones.

Mobility is a special case of concurrency, where the topology of the communication network is not fixed once for ever. In models for mobility, processes can create, pass and destroy communication links. An excellent introduction on these issues can be found in Milner's book [Mil99].

There are several models for concurrency, and in this course we will assume familiarity with one of them, namely *labelled transition systems*. For an overview of models for concurrency we refer to the Handbook chapter [WN95]. Also, we will assume that the reader is familiar with the notion of *bisimulation*, see [Mil99].

Probability is introduced in models for concurrency in essentially two ways. In the first kind of models, when a computation can perform different conflicting actions, the model provides a probability distribution over such actions. We call these models *probabilistic*. In the second kind of models, actions are assigned a duration. The length of the duration of an action is determined by a random variable. When a computation can perform different conflicting actions, the fastest is performed. We call these models *stochastic*. The theory of stochastic models is closer to the classic theory of stochastic processes, than the theory of probabilistic models is.

This minicourse will concentrate on probabilistic models.

2 Basic notions of probability

2.1 Measure spaces

Probability theory requires that we give probabilities to events. The probability of several mutually exclusive events should be the sum of the probabilities of the individual events. Also the total probability should sum up to 1. This brings about a difficulty that we are now going to explain.

Consider the set $[0, +\infty]$ of the extended nonnegative real numbers. Addition and multiplication are extended to $[0, +\infty]$ by

$$+\infty + x = +\infty; \quad +\infty \cdot x = \begin{cases} +\infty & \text{if } x > 0; \\ 0 & \text{if } x = 0. \end{cases}$$

Let X be a set and let $f : X \to [0, +\infty]$. For every $Y \subseteq X$ we define

$$f[Y] := \sum_{y \in Y} f(y) := \sup_{Z \subseteq_{fin} Y} \sum_{z \in Z} f(z)\,.$$

If X is a set of events, and if f assigns probabilities to the events in X, the number $f[Y]$ represents the probability of the totality of events in Y. The difficulty we mentioned above is expressed by the following proposition.

Proposition 2.1. *Let $f : X \to [0, +\infty]$ be a function, and let $Y := \{x \in X \mid f(x) > 0\}$ be the set of elements on which f is strictly positive (the* support *of f). If $f[Y] < +\infty$ then Y is at most countable.*

Proof: For every $n \in \mathbb{N}$ consider the set $Y_n := \{y \in Y \mid f(y) \geq \frac{1}{n}\}$. Clearly $|Y_n| \leq n \cdot f[Y]$, that is Y_n is finite. Notice that $Y = \bigcup_{n \in \mathbb{N}} Y_n$. Since the countable union of finite sets is at most countable, Y is at most countable. $\square$

This means that we cannot assign positive probabilities to uncountably many events, because the total sum would not be 1. However we often have to deal with uncountable spaces (like the real line for example, or the euclidean 3-dimensional space). To deal with such cases the notions of *measurable space* and of *measure* were introduced.

A *σ-algebra* on a set Ω is a family of subsets of X which is closed under countable union and complementation and which contains $\emptyset$. The intersection of an arbitrary family of σ-algebras is again a σ-algebra. In particular if $\mathcal{S} \subseteq \mathcal{P}(\Omega)$, and $\Xi := \{\mathcal{F} \mid \mathcal{F}$ is a σ-algebra & $\mathcal{S} \subseteq \mathcal{F}\}$ is the family of all σ-algebras that contain $\mathcal{S}$, then $\bigcap \Xi$ is again a σ-algebra and it belongs to Ξ. We call $\bigcap \Xi$ the *smallest* σ-algebra containing $\mathcal{S}$.

If $\mathcal{S}$ is a topology, the smallest σ-algebra containing $\mathcal{S}$ is called the *Borel* σ-algebra of the topology. Note that although a topology is closed under arbitrary union, its Borel σ-algebra need not be. A *measure space* is a pair $(\Omega, \mathcal{F})$ where $\mathcal{F}$ is a σ-algebra on Ω. Sets in $\mathcal{F}$ are called $\mathcal{F}$-measurable, or simply *measurable*. A measure space is *discrete* if $\mathcal{F} = \mathcal{P}(\Omega)$.

A *measure* on a σ-algebra $\mathcal{F}$ is a function $\nu : \mathcal{F} \to [0, +\infty]$ satisfying:

- (Strictness)
  $\nu(\emptyset) = 0$;
- (Countable additivity) if $(A_n)_{n \in \mathbb{N}}$ is a countable family of pairwise disjoint sets of $\mathcal{F}$, then
  $\nu(\bigcup_{n \in \mathbb{N}} A_n) = \sum_{n \in \mathbb{N}} \nu(A_n)$.

Finite additivity follows by putting $A_n = \emptyset$ for all but finitely many n. A measure is called a *probability* measure, when $\nu(\Omega) = 1$. It is called a *subprobability* measure, when $\nu(\Omega) \leq 1$. (Sub)probability measures over a discrete space are sometimes called (sub)probability *distributions*.

σ-algebras represent the sets over which it makes sense to give a notion of probability. Often, if we require the measure to satisfy some reasonable conditions, not all subsets of Ω can be made measurable.

Example: consider $[0, 1]$ with the euclidean topology. Define that the measure of every open interval is its length. This can be extended to a probability measure on the Borel σ-algebra. It is called the *Lebesgue* measure.

Another example: consider any set X and a countable subset $X_0 \subseteq X$. Define a function $f : X_0 \to [0, 1]$ such that $f[X_0] = 1$. Then the function $\nu(Y) = f[Y \cap X_0]$ is a discrete probability measure on X. The points of X_0 are the points where the "mass is concentrated". A special example are the measures where the mass is concentrated in only one point. These measures are called *Dirac* measures.

2.2 Markov chains

Markov chains are the simplest example of the notion of *stochastic process*. To introduce more general concepts we would need the introduction of the notion of *random variable*. This is not necessary in order to introduce Markov chains.

A (discrete, homogeneous) *Markov chain* is a pair $\langle X, p\rangle$, where X is a set (the state space) and $p : X \times X \to [0, 1]$ is a "stochastic matrix" that is for every $x \in X$

$$\sum_{y \in X} p(x, y) = 1 \; .$$

A Markov chain represents a process that changes state at every unit of time. The number $p(x, y)$ represents the probability that, given the fact that the process is now in state x, at the next unit of time it will be in state y.

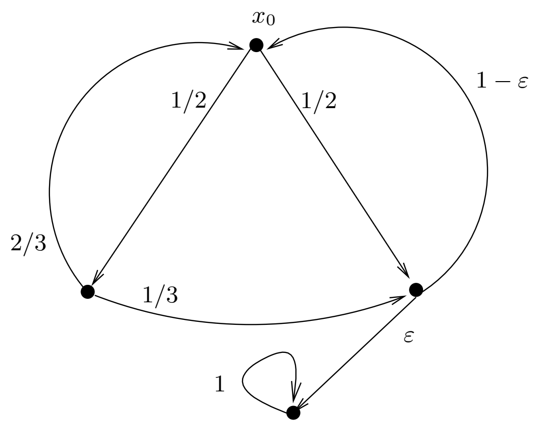


Figure 1: A Markov chain

The main feature of Markov chains is that the probability of the next state depends only on the present state, and it does not depend on the *history* of the process. "Given the present, the future is independent from the past": this is sometimes called a Markov property, or memorylessness. In the more general stochastic processes, the probability of the next state is allowed to depend on the history of the process, but these "memoryful" processes cannot be simply expressed using a stochastic matrix.

Markov chains can also be represented in the following way. Given a set X, let $V^1_\infty(X)$ be the set of discrete probability distributions over X.[1] A stochastic matrix can be then seen as a function

$$p : X \to V^1_\infty(X) \; .$$

[1]See the introduction of [Var03] if you want to know the reason of this notation

This representation is suitable to be generalised as we will see later.

We can endow a Markov chain with an *initial state* x_0. A *path* of such initialised Markov chain is a (finite of infinite) sequence of states $x_0x_1x_2x_3$.... The probability of a finite path is the product of all the $p(x_n, x_{n+1})$. We could define a probability on infinite paths as well, but in general every single infinite path will have probability 0. Therefore it is more interesting to define a measure space on the set Ω of infinite paths.

Let τ be a finite path. The *shadow* $K(\tau)$ is the set of all infinite paths that extend τ. We define $\mathcal{F}$ to be the σ-algebra generated by the shadows of finite paths. Using standard

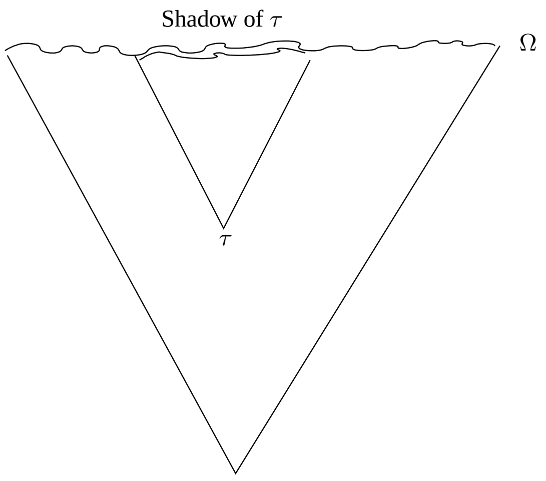


Figure 2: The set of paths

measure theory results, one can show that there exists a unique probability measure ν on $\mathcal{F}$ such that for every finite path τ, the probability of τ as described above coincides with $\nu(K(\tau))$.

Now any reasonable event that can be described in words defines a set of $\mathcal{F}$ and it can be assigned a probability.

Example: random walk. The state space is $\mathbb{Z}$, the set of positive and negative integers. The stochastic matrix is given by $p(n,m) = 1/2$ if $m = n+1$ or $m = n-1$, and $p(n,m) = 0$ otherwise. Let's also put 0 as the initial state. Paths are constituted by sequences of numbers. We can restrict our attention to sequences of *adjacent* numbers, as every other path has probability 0.

What is the probability that an infinite path hits 0 infinitely many times? It can be shown that the set of paths for which 0 appears infinitely often is measurable, and that its measure is 1.

3 Probabilistic LTSs

We give a uniform presentation of probabilistic models. This section is based on [BSdV03].

Let $\mathcal{P}(X)$ represent the powerset of X, while by $V(X)$ we will denote the set of discrete *sub*probability distributions. [2] Note the difference with the previous section, where we only consider probability distributions. We don't always need this extra generality, but it helps giving an homogeneous presentation. The missing probability is usually interpreted as the probability of blocking.

We saw that a Markov chain can be seen as a function

$$X \to V(X) \, .$$

[2]It should be $V_{\infty}^{\leq 1}(X)$ to be precise.

We will present all our probabilistic models in a similar way, following what it is called the "coalgebraic approach" . This name comes from the fact that in category theory, objects of the form $X \to F(X)$ for some endofunctor F are called *coalgebras*. We will not enter in any detail of the category theory underlying this approach. It is worth mentioning, though, that coalgebras are equipped with a natural notion of bisimulation. Presenting models in a coalgebraic way induces automatically a definition of bisimulation.

Labelled transition systems (LTS) can also be presented coalgebraically. Let A be a set of labels. Then an LTS with labels A is given by a set of states X and a transition relation $t \subseteq X \times A \times X$. The transition relation can alternatively be seen as a function

$$t : X \times A \to \mathcal{P}(X) ,$$

which can also be written (by "currying") as

$$t : X \to (A \to \mathcal{P}(X)) .$$

In this way the LTS is seen as a process that, when it is in a state x, and after receiving in input a label a, nondeterministically chooses to enter in one of the states in $t(x)(a)$. This is the *reactive* view of an LTS. Deterministic automata are a special case, where the transition function is in fact: $t : X \to (A \to X)$.

We can also write the transition as a function

$$t : X \to \mathcal{P}(A \times X) .$$

In this way the LTS is seen as a process that, when it is in a state x, nondeterministically outputs a label and enters in a new state. This is the *generative* view of an LTS.

The above discussion shows that in a LTS the notion of input and output are not defined. This is reflected in process languages like CCS, where the synchronization is completely symmetric, and the distinction between the roles of emitter and receiver is irrelevant.

Let us now substitute nondeterministic choice with probabilistic choice. This amounts to writing V in place of $\mathcal{P}$. The first probabilistic model we obtain is given by a set of states X together with a transition function

$$t : X \to (A \to V(X)) .$$

This model represents a process that, when it is in a state x, and after receiving in input a label a, enters the state y with probability $t(x)(a)(y)$. We call such a model a *reactive (probabilistic) LTS*. They are also known as labelled Markov processes (and in fact they are called in many other ways!).

The second probabilistic model is given by a set of states X together with a transition function

$$t : X \to V(A \times X) .$$

This model represents a process that, when it is in a state x, emits the label a and enters the state y with probability $t(x)(a, y)$. We call such a model a *generative (probabilistic) LTS*. Unlike the nondeterministic case, generative LTSs and reactive LTSs are not two aspects of the same notion. In both cases the transition function can be seen as a function $t : X \times A \times X \to [0, 1]$ but in a generative LTS we have that for every $x \in X$

$$\sum_{a \in A, y \in X} t(x, a, y) \leq 1 ,$$

while in a reactive LTS we have that for every $x \in X$ and *for every* $a \in A$

$$\sum_{y \in X} t(x, a, y) \leq 1 .$$

In probabilistic systems it is important to know who is outputting and who is inputting.

But this is not the whole story. The next probabilistic model we present is given by a set of states X together with a transition function

$$t : X \to (A \to \mathcal{P}(V(X))) \; .$$

This model represents a process that, when it is in a state x, and after receiving a label a, nondeterministically chooses a probability distribution ν in $t(x)(a)$ and then enters in state y with probability $\nu(y)$. We call this model a *simple Segala automaton*. (This model, or slight variations of it, is known under very many other different names.) Reactive systems are a special case of simple Segala automata, where for every x and a the set $t(x)(a)$ is a singleton.

Note that in Segala automata, the symmetry is regained. In fact, a Segala automaton in the "reactive" form as above is equivalent to a more "generative" version where the transition function is of the form

$$t : X \to \mathcal{P}(A \times V(X)) \; .$$

This can be seen as a process that nondeterministically chooses a label and a probability distribution over next states. We can generalise even more the model to obtain the *generalised Segala automaton*. This is given by a set of states X together with a transition function

$$t : X \to \mathcal{P}(V(A \times X)) \; .$$

This can be seen as a process that nondeterministically chooses a probability distribution over labels and next states. The combination of probability and nondeterminism is an important factor. For example it is essential in order to give a semantics to the probabilistic π calculus.

4 Reactive systems

Recall that a reactive system S is a pair $\langle X, t : X \to (A \to V(X))\rangle$. To simplify the notation we consider the transition function as $t : A \to (X \to V(X))$ and we denote $t(a)$ as t_a. For every $x \in X, a \in A$, $t_a(x)$ is a subprobability distribution over X. The most important instances are probability distributions ($t_a(x)[X] = 1$) and "null functions" ($t_a(x)[X] = 0$). In the second case we say that the label a is *not enabled* at x.

A reactive system is *initialised* when an initial state x_0 is specified.

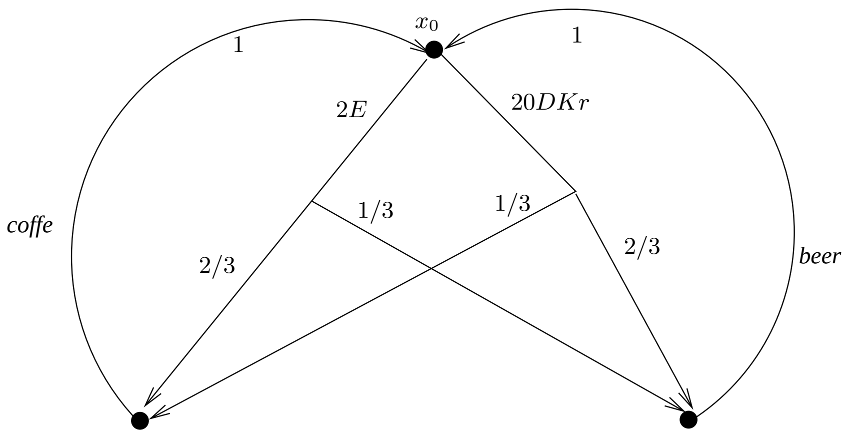


Figure 3: A reactive vending machine

Figure 3 represents a reactive vending machine that is more likely to give you beer if you use Danish Kroner. In the picture, if a label a does not appear in any edge with source x, it means that a is not enabled at x.

4.1 Bisimulation and logic

The main equivalence relation on reactive LTSs is that of *bisimilarity*, defined by means of the notion of *bisimulation*.

A bisimulation between two reactive systems S, S' is an equivalence relation R on $X \cup X'$ such that whenever xRx', for every equivalence class C of $X \cup X'/R$ and for every label a:

$$t_a(x)[C] = t'_a(x')[C]\ .$$

Two states are bisimlar when there is a bisimulation relating them. Two initialised processes are bisimlar when their initial states are.

Roughly speaking, two states are bisimilar when for every label, the corresponding probabilities of reaching bisimilar states are the same.

We can characterise bisimilarity using a logic, that we call the *Larsen-Skou* logic. This logic is a generalisation of the Hennessy-Milner logic that characterises bisimilarity for nonprobabilistic LTSs.

The formulas of the Larsen-Skou logic are as follows:

$$\varphi ::= \mathbf{T} \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \langle a\rangle_q \varphi$$

The semantics of the formulas is given in terms of a satisfaction relation

$$x \vDash_S \varphi$$

Where x is a state of the reactive system S. The semantic of the boolean combinators is obvious. To express the semantics of the “modal” operator, let $[\![\varphi]\!]_S$ denote the set of states of S that satisfy the formula φ. Then we say that

$$x \vDash_S \langle a\rangle_q \varphi$$

whenever

$$t_a(x)[\![\varphi]\!]_S > q\ .$$

The Hennessy-Milner logic is a special case, where simply $q = 0$ in every formula. In the nonprobabilistic case, two processes are bisimilar if and only if they satisfy the same formulas of the Hennessy-Milner logic. Similarly to the nonprobabilistic case we have the following result:

Theorem 4.1. *Two states of a reactive system are bisimilar if and only if they satisfy the same formulas of the Larsen-Skou logic.*

The interesting part is that this result is still true for a logic *without negation*. The proof involves the use of uncountable spaces, and measure theory. This fact is very important, so we stress it

Theorem 4.2. *Two states of a reactive system are bisimilar if and only if they satisfy the same formulas of the Larsen-Skou logic without negation.*

4.2 Approximate bisimulation and metrics

Bisimulation is a very fine equivalence. A slight difference in the probability distributions makes two processes non bisimilar. However precise numbers in this framework make little sense: they should be intended as estimates. Real numbers can never be exactly computed anyway.

These considerations suggest the use of a more flexible notion than the one of exact equivalence: *approximate equivalence*. The formal way to define this is via the notion of *metrics*.

A metrics on a set X is a function specifying, for every pair of elements of X, their relative distance. It has to satisfy some reasonable axioms that we are now going to present.

A *pseudo-metrics* on a set X is a function $d : X \times X \to [0, +\infty]$ which satisfies:

- $d(x, x) = 0$ (reflexivity)
- $d(x, y) = d(y, x)$ (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangular inequality)

If the pseudo-metrics satisfies also $d(x, y) = 0 \Longrightarrow x = y$ then it is called a *metrics*.

Given a pseudo-metrics d, we can define an equivalence relation R by saying that $(x, y) \in R$ if $d(x, y) = 0$ — if they live in the same "place".[3] The aim is to define a pseudo-metrics between (states of) processes, with the property the $d(x, y) = 0$ exactly when x and y are probabilistically bisimilar.

To define this distance we use the logics. Formulas before were valued in $\{0, 1\}$ (either they are satisfied or they aren't). We will now make them take values in $[0, 1]$. The new formulas are interpreted as functions $f : X \to [0, 1]$. They are as follows

$$\begin{aligned} f \quad ::= \quad & \mathbf{1} \mid \mathbf{1} - f \mid \min(f_1, f_2) \mid \sup_{i \in I} f_i \\ & f \ominus q \mid \langle a \rangle f \end{aligned}$$

Given a reactive system $S = \langle X, t : A \times X \to V(X) \rangle$, and fixed a constant $0 < c < 1$, these expressions define functions $X \to [0, 1]$ as follows:

- $\mathbf{1}(x) = 1$;
- $f \ominus q(x) = \max(f(x) - q, 0)$;
- $\langle a \rangle f(x) = c \sum_{y \in X} t_a(x)(y) f(y)$;

while $\min$, $\sup$, and $\mathbf{1} - f$ are defined in the obvious way.

The constant c discounts the future behaviour. The distance between two states is defined as the "Kantorowich" metrics:

$$d(x, y) = \sup_f |f(x) - f(y)|$$

It can be proved that this function d is a pseudo-metrics on the set of states, and that $d(x, y) = 0$ if and only if x, y are bisimilar.

The modal operator of the Larsen-Skou logic is split here into two parts: we have a translation $\varphi \mapsto f$ such that

$$x \vDash_S \varphi \Leftrightarrow f(x) > 0$$

with $\langle a \rangle_q \varphi \mapsto \langle a \rangle f \ominus q$.

Given this translation, and the logical characterisation of bisimilarity given above, it is intuitive (although not automatic) that $d(x, y) = 0$ if and only if x and y are bisimilar.

The exact value of the distance depends on the constant c. However the constant c is not important in the sense that different choices for it produce the same topology (in fact the same "uniformity"). Roughly speaking they define the same notion of "being closer": the inequality $d(x, y) \leq d(x', y')$ does not depend on c.

This distance compares the initial behaviours of systems. If two systems differ at the beginning, they are far apart. It is possible to define a distance that compares the eventual

[3] Like Qfwfq, de XueaeuX and Mrs. Ph(i)nk$_0$ - Italo Calvino, "All at one point", from the collection "Cosmicomics", 1965.

infinite behaviour, in such a way that systems that are very different at the start, can be very close to each other at the limit.

Other metrics, and other notions of approximate bisimilarity are defined in the literature.

5 Generative systems

An (initialised) generative probabilistic LTS is constituted by $\langle X, x_0, t : X \to V(A \times X)\rangle$, where x_0 is the initial state.

Generative systems were used to give semantics to a language called "probabilistic synchronous CCS". Given a set of atomic actions $Atom$, its syntax is the following

$$P ::= a.P \mid P +_p Q \mid P \times Q \mid rec_X P \mid X$$

where $a \in Atom$. The operational semantics is given in terms of generative systems on the set of labels $A = Atom^*$, the finite strings of actions in $Atom$.

$$a.P \xrightarrow[1]{a} P$$

$$\frac{P \xrightarrow[q]{\sigma} P'}{P +_p Q \xrightarrow[pq]{\sigma} P'}$$

$$\frac{Q \xrightarrow[q]{\sigma} Q'}{P +_p Q \xrightarrow[q(1-p)]{\sigma} Q'}$$

$$\frac{P \xrightarrow[p]{\sigma} P' \qquad Q \xrightarrow[q]{\tau} Q'}{P \times Q \xrightarrow[pq]{\sigma\tau} P' \times Q'}$$

and the standard rule for recursion.

It is not difficult to prove that the above rules produce a generative system. By imposing commutativity of the composition in $Atom^*$, we can also make the operator $\times$ commutative.

6 Segala automata

Segala automata were introduced by Segala in is PhD thesis [Seg95] under the supervision on Nancy Lynch. A recent presentation in [Sto02]. Similar models can be found elsewhere, see the bibliographic section.

6.1 Notation

A generalised Segala automaton is given by a set of states X together with a transition function

$$t : X \to \mathcal{P}(V(A \times X)) .$$

Simple Segala automata are a special case, where every (sub)probability distribution concerns only one label, so that the transition function can be seen as

$$t : X \to \mathcal{P}(A \times V(X)) .$$

Sometimes a special blocking state $\perp$ is chosen so that the transition function is in fact

$$t : X \to \mathcal{P}(A \times (V(X) \cup \{\perp\}) \,.$$

Initialised automata are automata with a special initial state x_0.

In the following, for simplicity we will consider only automata without blocking state, and only subprobability distributions that are in fact probability distributions.

The notation we use comes from [HP00]. Consider a transition function t. Whenever a probability distribution ν belongs to $t(x)$ for a state $x \in X$ we will write

$$x\{ \xrightarrow[p_i]{a_i} x_i\}_{i\in I}$$

where $x_i \in X$, $i \neq j \implies (a_i, x_i) \neq (a_j, x_j)$, and $\nu(a_i, x_i) = p_i$.

A good way of visualising probabilistic automata is by using alternating graphs [Han91], see Figure 4. Black nodes represent states, hollow nodes represent probability distributions.

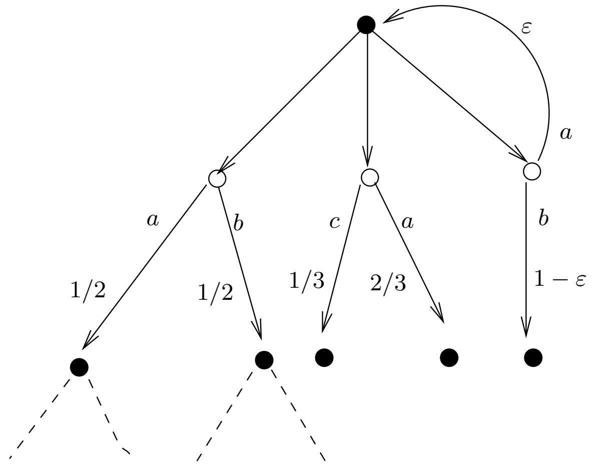


Figure 4: A Segala automaton

In a simple Segala automaton we can move the labels to the arcs outgoing a black node, see Figure 5.

An important notion that we will need is that of (extended) *convex combination* of (sub)probability distributions.

Given a set of probability distributions D on a set X, we define the set $\overline{D}$ to be the set of probability distributions ν on X such that there exist $\nu_i \in D$ and $p_i \in [0, 1]$, for $i \in I$ satisfying:

- $\sum_{i\in I} p_i = 1$;
- for all $x \in X$, $\nu(x) = \sum_{i\in I} p_i\nu_i(x)$.

A convex combination of probability distributions represents, in some sense, a probability distribution over probability distributions. To excite the curiosity of the reader I can say that this corresponds to the multiplication of the probabilistic monad. See [Var03] for more details.

6.2 Paths and schedulers

A *finite path* of an (intialised) Segala automaton is an element in $(X \times V(X \times A) \times A)^*X$, written as $x_0\nu_1a_1x_1 \ldots \nu_na_nx_n$. An *infinite* path is defined in a similar way as an element

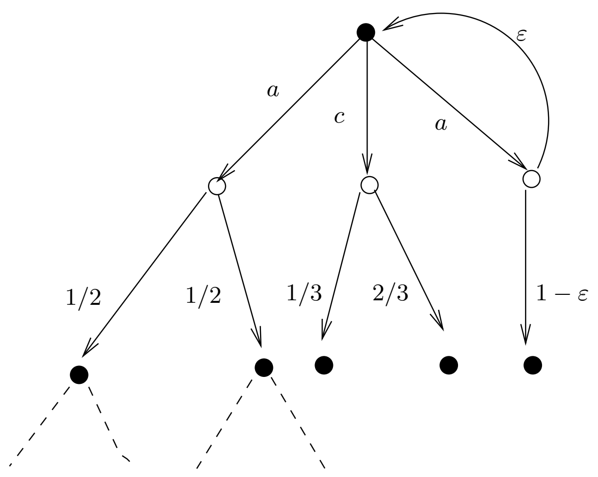


Figure 5: A simple Segala automaton

of $(X \times V(X \times A) \times A)^\omega$. The path is *deterministic* if $\nu_{i+1} \in t(x_i)$. It is *probabilistic* if $\nu_{i+1} \in \overline{t(x_i)}$.

The probability of a path $\tau := x_0\nu_1a_1x_1 \ldots \nu_na_nx_n$ is defined as

$$\Pi(\tau) = \prod_{1\leq i\leq n} \nu_i(a_ix_i)\,.$$

The last state of a finite path τ is denoted by $l(\tau)$. A path τ is *maximal* if it is infinite or if $t(l(\tau)) = \emptyset$.

A *probabilistic scheduler* for a probabilistic automaton with transition function t is a partial function $\mathcal{S} : (X \times V(X \times A) \times A)^*X \to V(X \times A)$ such that

- if $t(l(\tau)) \neq \emptyset$ then $\mathcal{S}(\tau)$ is defined;
- $\mathcal{S}(\tau) \in \overline{t(l(\tau))}$.

Equivalently, a probabilistic scheduler can be defined as a partial function $\mathcal{S} : (X \times V(X \times A) \times A)^*X \to V(V(X \times A))$, requiring that $\mathcal{S}(\tau)(\nu) > 0 \Longrightarrow \nu \in t(l(\tau))$.

A *deterministic* scheduler is a probabilistic scheduler that does not make use of the convex combinations. That is for a deterministic scheduler we have $\mathcal{S}(\tau) \in t(l(\tau))$.

A deterministic scheduler chooses the next probability distribution, knowing the history of the process. Using the representation with alternating graphs, we can say that, for every path ending in a black node, a scheduler chooses one of his hollow sons. A probabilistic scheduler chooses a convex combination of distributions, which corresponds to putting a probability distribution over probability distributions. Schedulers can also decide to ignore the history, in such a case we talk of memoryless schedulers. For every black node,a deterministic memoryless scheduler chooses one of his hollow sons.

Now, given an (initial) state $x_0 \in X$ and a scheduler $\mathcal{S}$ for t, we consider the set $\mathcal{B}(t, x_0, \mathcal{S})$ of maximal paths, obtained from t by the action of $\mathcal{S}$. Those are the paths $x_0\nu_1a_1x_1 \ldots \nu_na_nx_n$ such that $\nu_{i+1} = \mathcal{S}(x_0\nu_1a_1x_1 \ldots \nu_ia_ix_i)$. A deterministic scheduler produces deterministic paths, a probabilistic scheduler produces probabilistic paths.

The set $\mathcal{B}(t, x_0, \mathcal{S})$ of maximal paths obtained by a scheduler $\mathcal{S}$ is endowed with the σ-algebra generated by finite paths, in the similar way as seen for Markov chains. Let $\mathcal{X} := \{K(\tau) \mid \tau \text{ finite}\}$ be the set of shadows of finite paths, let $\mathcal{F}$ be the smallest σ-algebra containing $\mathcal{X}$. Let $v : \mathcal{X} \to [0, 1]$ be defined as $v(K(\tau)) := \Pi(\tau)$. It can be proved that v extends to a unique probability measure ν on $\mathcal{F}$.

If we are interested in the labels only, we can remove states and distributions from the paths and get a probability measure over the set of sequences A^ω. This procedure is more easily understood when applied to finite paths.

The finite paths semantics is given in terms of "probabilistic languages": given an alphabet A, a *probabilistic word* of length n over A is a probability distribution over strings in A^n. A *probabilistic language* is a set of probabilistic words. Given a length n, a scheduler $\mathcal{S}$ for a Segala automaton t produces a probabilistic word of length n as follows. Consider the paths in $\mathcal{B}(t, x_0, \mathcal{S})$ that have length n with their probability. Let $erase(\tau)$ be the string in A^n obtained from τ by erasing states and probability distributions. Given such a string $\rho \in A^n$, its probability is the sum of the probabilities of the paths that produce it.

$$\Pi(\rho) := \sum_{erase(\tau)=\rho} \Pi(\tau)$$

This induces a probabilistic word. Given a class of schedulers for the automaton, its language is the set of probabilistic words obtained by the schedulers in the class.

6.3 A verification logic

The logic PCTL has the following syntax. It's divided in path formulas and state formulas.

$$\begin{array}{rcl}
\text{State formulas } \varphi & ::= & \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \mid \alpha \\
 & & \exists_{>p}\psi \mid \forall_{>p}\psi \\
\text{Path formulas } \psi & ::= & X\varphi \mid \varphi U \varphi
\end{array}$$

The semantics of the formulas is the following (consider only simple Segala automata for simplicity). For state formulas we define a satisfaction semantics

$$x \vDash \varphi$$

where x is a state of the automaton. We have

$$x \vDash \langle a \rangle$$

if $x\{ \xrightarrow[p_i]{a} x_i\}_{i \in I}$. More generally we can have "atomic" propositions α such that the set of states x for which

$$x \vDash \alpha$$

is assigned in some way, which is not relevant here (α can represent "internal" properties of a state).

For path formulas we define

$$\tau \vDash \psi$$

where τ is a path of the automaton. Suppose $\tau = x_0\nu_1 a_1 x_1 \dots \nu_n a_n x_n$, then we have that

$$\tau \vDash X\varphi$$

if $x_1 \vDash \varphi$. We have

$$\tau \vDash \varphi_1 U \varphi_2$$

if there is i such that $x_i \vDash \varphi_2$ and for all $j < i$, $x_j \vDash \varphi_1$. So far the semantics is very similar to the classic semantics of CTL. Now for the two "probabilistic" operators, we have that

$$x \vDash \exists_{>p}\psi$$

if there exists a scheduler $\mathcal{S}$ for t, x such that the set of paths $\tau \in \mathcal{B}(t, x_0, \mathcal{S})$ such that $\tau \vDash \psi$ has measure $> p$. Dually

$$x \vDash \forall_{>p}\psi$$

if for all schedulers $\mathcal{S}$ for t, x the set of paths $\tau \in \mathcal{B}(t, x_0, \mathcal{S})$ such that $\tau \vDash \psi$ has measure $> p$. Of course, in order for this definition to make sense, one has to check first that the sets described above are measureable. This is indeed the case.

PCTL is used in probabilistic verification. This involves several algorithmic issues that are beyond the scope of this course.

7 Probabilistic π calculus

This section is taken from [HP00]. We assume the reader is familiar with the π-calculus [Mil99].

The probabilistic asynchronous π-calculus has the following syntax:

$$\begin{array}{rrcl} \text{Prefixes} & \alpha & ::= & x(y) \mid \tau \\ \text{Processes} & P & ::= & \mathbf{0} \mid \bar{x}y \mid \sum_i p_i\alpha_i.P_i \mid P\|P \mid \nu x P \mid X \mid rec_X P \end{array}$$

Similarly to the asynchronous π-calculus, output actions $\bar{x}y$ cannot have a continuation. This represents the asynchrony of messages: the sender does not wait for the message to be received.

The difference with usual π-calculus is that the nondeterministic sum is replaced by the probabilistic sum $\sum_i p_i\alpha_i.P_i$, where $p_i > 0$ and $\sum_i p_i = 1$.

The semantic of π can be given in terms of generalised Segala automata over labels of the form

$$\lambda ::= x(y) \mid \bar{x}y \mid \nu y\bar{x}y \mid \tau$$

where $x(y)$ represents input of a name y along channel x, $\bar{x}y$ represents output of the name y along channel x, $\nu y\bar{x}y$ represents output of a fresh name, and τ represents a silent action. (For the details of the labelled transition semantics of π, ask your nearest π-calculus guru.)

$$\textsf{SUM}\quad \textstyle\sum_i p_i\alpha_i.P_i\{\xrightarrow[p_i]{\alpha_i} P_i\}_i$$

$$\textsf{OUT}\quad \bar{x}y\{\xrightarrow[1]{\bar{x}y} \mathbf{0}\}$$

$$\textsf{OPEN}\quad \frac{P\{\xrightarrow[1]{\bar{x}y} P'\}}{\nu y P\{\xrightarrow[1]{\nu y\bar{x}y} P'\}}\quad x \neq y$$

$$\textsf{PAR}\quad \frac{P\{\xrightarrow[p_i]{\lambda_i} P_i\}_i}{P\|Q\{\xrightarrow[p_i]{\lambda_i} P_i\|Q\}_i}$$

$$\textsf{RES}\quad \frac{P\{\xrightarrow[p_i]{\lambda_i} P_i\}_i}{\nu y P\{\xrightarrow[p'_i]{\lambda_i} \nu y P_i\}_{i:y\notin fn(\lambda_i)}}$$

Where p'_i is a probability, renormalised over the set of labels whose free names are not restricted[4]

$$p'_i = \frac{p_i}{\sum_{j:y\notin fn(\lambda_j)} p_j}$$

[4] A name is not free only if it is the object of an input or of a fresh output.

Then we have the two communication rules. In order to introduce them, we need the notation

$$\{\xrightarrow[p_i]{\lambda_i} P_i\}_{i\in I} \oplus \{\xrightarrow[p_j]{\lambda_j} P_j\}_{j\in J}$$

which represents

$$\{\xrightarrow[p_k]{\lambda_i} P_k\}_{k\in I\uplus J}$$

where $I \uplus J$ denotes the disjoint union.

$$\text{COM}\ \frac{P\{\xrightarrow[1]{\bar{x}y} P'\} \qquad\qquad Q\{\xrightarrow[p_i]{\lambda_i} Q_i\}_i}{P\|Q\{\xrightarrow[p_i]{\tau} P'\|Q_i[y/z_i]\}_{i:\lambda_i=x(z_i)} \oplus \{\xrightarrow[p_i]{\lambda_i} P'\|Q_i\}_{i:\lambda_i\neq x(z_i)}}$$

$$\text{CLOSE}\ \frac{P\{\xrightarrow[1]{\nu y\bar{x}y} P'\} \qquad\qquad Q\{\xrightarrow[p_i]{\lambda_i} Q_i\}_i}{P\|Q\{\xrightarrow[p_i]{\tau} \nu y(P'\|Q_i[y/z_i])\}_{i:\lambda_i=x(z_i)} \oplus \{\xrightarrow[p_i]{\lambda_i} P'\|Q_i\}_{i:\lambda_i\neq x(z_i)}}$$

The two rules differ only in whether the communicated name is fresh or not. This issue is inherited from the π calculus. The intuitive behaviour expressed by this rule is the following: when P and Q decide to synchronise on x, such synchronisation happens “maximally”, that is for every input action of Q whose subject channel is x. If Q is listening over other channels, or doing other τ transitions, those actions are not affected. The last rules are the standard congruence rule, where the structural congruence $\equiv$ is defined as in the π-calculus (so as to make parallel composition commutative, for instance)

$$\text{CONG}\ \frac{P\equiv P' \qquad P'\{\xrightarrow[p_i]{\lambda_i} Q'_i\}_i \qquad Q_i\equiv Q'_i}{P\{\xrightarrow[p_i]{\lambda_i} Q_i\}_i}$$

and the standard rule for recursion.

8 Event structures

A different approach to concurrency is represented by the so called *causal* models, where information of causality and concurrency is recorded. One of such models is the one we present here.

An *event structure* is a triple $\mathcal{E} = \langle E, \leq, \#\rangle$ such that

- E is a countable set of *events*;
- $\langle E, \leq\rangle$ is a partial order, called the *causal order*, such that for every $e \in E$, the set of events $\downarrow e$ is finite;
- $\#$ is an irreflexive and symmetric relation, called the *conflict relation*, satisfying the following: for every $e_1, e_2, e_3 \in E$ if $e_1 \leq e_2$ and $e_1\#e_3$ then $e_2\#e_3$.

We say that the conflict $e_2\#e_3$ is *inherited* from the conflict $e_1\#e_3$, when $e_1 < e_2$. Causal dependence and conflict are mutually exclusive. If two events are not causally dependent nor in conflict they are said to be *concurrent*.

A *configuration* x of an event structure $\mathcal{E}$ is a conflict-free downward closed subset of E, i.e., a subset x of E satisfying (1) whenever $e \in x$ and $e' \leq e$ then $e' \in x$ and (2) for every $e, e' \in x$, it is not the case that $e\#e'$. Therefore, two events of a configuration are either causally dependent or concurrent, i.e., a configuration represents a run of an event structure where events are partially ordered. The set of configurations of $\mathcal{E}$, partially ordered by inclusion, is denoted as $\mathcal{L}(\mathcal{E})$. The set of finite configurations is written by $\mathcal{L}_{fin}(\mathcal{E})$. We denote the empty configuration by $\perp$. If x is a configuration and e is an event such that $e \notin x$ and $x \cup \{e\}$ is a configuration, then we say that e is *enabled* at x. Two configurations x, x' are said to be *compatible* if $x \cup x'$ is a configuration. For every event e of an event structure $\mathcal{E}$, we define $[e] := \downarrow e$, and $[e) := [e] \setminus \{e\}$. It is easy to see that both $[e]$ and $[e)$ are configurations for every event e and that therefore any event e is enabled at $[e)$.

We say that events e_1 and e_2 are in *immediate* conflict, and write $e_1 \#_\mu e_2$ when $e_1 \# e_2$ and both $[e_1) \cup [e_2]$ and $[e_1] \cup [e_2)$ are configurations. Note that the immediate conflict relation is symmetric. It is also easy to see that a conflict $e_1 \# e_2$ is immediate if and only if there is a configuration where both e_1 and e_2 are enabled. Every conflict is either immediate or inherited from an immediate conflict.

8.1 Confusion-free event structures

The most intuitive way to add probability to an event structure is to identify "probabilistic events", such as coin flips, where probability is associated locally. A probabilistic event can be thought of as probability distribution over a *cell*, that is, a set of events (the outcomes) that are pairwise in immediate conflict and that have the same set of causal predecessors. The latter implies that all outcomes are enabled at the same configurations, which allows us to say that the probabilistic event is either enabled or not enabled at a configuration.

Definition 8.1. A *partial cell* is a non-empty set c of events such that $e, e' \in c$ implies $e \#_\mu e'$ and $[e) = [e')$. A maximal partial cell is called a *cell*.

We will now restrict our attention to event structures where each immediate conflict is resolved through some probabilistic event. That is, we assume that cells are closed under immediate conflict. This implies that cells are pairwise disjoint.

Definition 8.2. An event structure is *confusion-free* if its cells are closed under immediate conflict.

If this is the case, cells are the equivalence classes of the reflexive closure of immediate conflict. In a confusion-free event structure, if an event $e \in c$ is enabled at a configuration x, all the events of c are enabled as well.

Once an event structure is confusion-free, we can associate a probability distribution with each cell. Intuitively it is as if we have a die local to each cell, determining the probability with which the events at that cell occur.

Recall that when $f : X \to [0, +\infty]$ is a function, for every $Y \subseteq X$, we define $f[Y] := \sum_{x \in Y} f(x)$.

Definition 8.3. A *cell valuation* on a confusion-free event structure $\langle E, \leq, \# \rangle$ is a function $p : E \to [0, 1]$ such that for every cell c, we have $p[c] = 1$. A *probabilistic event structure* is an event structure together with a cell valuation.

Assuming probabilistic independence of all probabilistic events, every finite configuration can be given a "probability" which is obtained as the product of probabilities of its constituent events. This gives us a weight function $\mathcal{L}_{fin}(\mathcal{E}) \to [0, 1]$ defined by $v(x) = \Pi_{e \in x} p(e)$. This weight function defines a probability measure on the set of maximal configurations $\Omega(\mathcal{L}(\mathcal{E}))$:

For every finite configuration x the set $K(x) := \uparrow x \cap \Omega(\mathcal{L}(\mathcal{E}))$ is called the *shadow* of x. We shall consider the σ-algebra $\mathcal{S}$ on generated by the shadows of the compact elements.

Theorem 8.4. *Let v be weighting function as defined above. Then there is a unique probability measure μ on $\mathcal{S}$ such that for every finite configuration x, $\mu(K(x)) = v(x)$.*

Independence of cells means the following: let μ be a probability measure on $\mathcal{S}$ that is generated by a cell valuation. Then every two configurations are probabilistically independent given the common past

Theorem 8.5. *let μ be a probability measure on $\mathcal{S}$ that is generated by a cell valuation. Then for every two finite compatible configurations x, y*

$$\mu\Big(K(x) \cap K(y) \mid K(x \cap y)\Big) = \mu\Big(K(x) \mid K(x \cap y)\Big) \cdot \mu\Big(K(y) \mid K(x \cap y)\Big) .$$

(The statement of this theorem requires some more probability theory than that we have presented. Exercise: understand it!)

What is the meaning of lack of independence? Read [VVW04] :-)

8.2 Finite runs

What is the probabilistic version of finite configurations? If probabilistic words are probability distributions over words, probabilistic finite configurations should be probability distribution over finite configurations. But which sets are suitable to be the support of such distribution? In the case of Segala automata, the sets of runs of the same length do the job. For event structures this won't do.

To see why consider the event structure with only two concurrent events a, b. The only maximal run assigns probability 1 to the maximal configuration $\{a, b\}$. This corresponds to a configuration valuation which assigns 1 to both $\{a\}$ and $\{b\}$. Now these are two configurations of the same size, but their common "probability" is equal to 2. Not a probability! The reason is that the two configurations are compatible: they do not represent *alternative* choices. We therefore need to represent alternative choices, and we need to represent them all. This leads us to the following definition.

Definition 8.6. Let $\mathcal{E}$ be an event structure. A *partial test* of $\mathcal{E}$ is a set C of pairwise incompatible configurations of $\mathcal{E}$. A *test* is a maximal partial test. A test is *finitary* if all its elements are finite.

Maximality of a partial test C can be characterised equivalently as *completeness*: for every maximal configuration z, there exists $x \in C$ such that $x \subseteq z$. The set of tests is naturally endowed with the Egli-Milner order: $C \leq C'$ if and only if

- for every $x \in C$ there exists $x' \in C'$ such that $x \subseteq x'$;
- for every $x' \in C'$ there exists $x \in C$ such that $x \subseteq x'$.

Tests were designed to support probability distributions. They indeed do, as the following result shows.

Definition 8.7. Let v be a function $\mathcal{L}_{fin}(\mathcal{E}) \to [0, 1]$. Then v is called a *test valuation* if for all finitary tests C we have $v[C] = 1$.

Theorem 8.8. *Let v be a weighting function generated by a cell valuation. Then v is a test valuation.*

8.3 A process language

Confusion-freeness is a strong requirement. But it is still possible to give a semantics to a fairly rich language for probabilistic processes in terms of probabilistic event structures with independence. The language we sketch is a probabilistic version of value passing CCS.

Assume a set of channels A, and a set of values V that can be communicated over any channel $a \in A$. The syntax of processes is given by:

$$\begin{aligned} P \quad ::= \quad & 0 \mid \sum_{v \in V} a!(p_v, v).P_v \mid a?(x).P \mid P_1 \| P_2 \mid \\ & \textbf{if } b \textbf{ then } P_1 \textbf{ else } P_2 \mid X \mid rec_X P \end{aligned}$$

Here x ranges over value variables and b over boolean expressions (which make use of values and value variables). The coefficients p_v are real numbers such that $\sum_{v \in V} p_v = 1$.

A closed process denotes a probabilistic event structure with an additional labelling function from events to output labels $a!v$, input labels $a?v$ where a is a channel and v a value, or τ. We sketch the probabilistic semantics in terms of the non probabilistic semantics of CCS. (See e.g. the handbook chapter [WN95] for an explanation of the event structure semantics of CCS.)

The nil process 0 denotes the empty probabilistic event structure. A closed output process $\sum_{v \in V} a!(p_v, v).P_v$ outputs a value v on a channel a with probability p_v, and then continues as the process P_v. Each P_v, for $v \in V$, will denote a labelled probabilistic event structure $\mathcal{E}[\![P_v]\!]$. The event structure denoted by $\sum_{v \in V} a!(p_v, v).P_v$ is the juxtaposition of the family of prefixed event structures

$$a!v.\mathcal{E}[\![P_v]\!] \; ,$$

with $v \in V$, in which the additional prefixing events labelled $a!v$ are put in (immediate) conflict; the new prefixing events labelled $a!v$ are then assigned probabilities p_v.

A closed input process $a?(x).P$ inputs a value v on channel a and resumes as the closed process $P[v/x]$. Such a process $P[v/x]$ denotes a labelled probabilistic event structure $\mathcal{E}[\![P[v/x]]\!]$. The event structure of the input process is got as the parallel juxtaposition of the family of prefixed event structures

$$a?v.\mathcal{E}[\![P[v/x]]\!] \; ,$$

with $v \in V$; each new prefixing event labelled $a?v$ is then assigned probability 1.

The probabilistic parallel composition corresponds to the usual CCS parallel composition followed by restricting away on all channels used for communication. In order for the parallel composition $P_1 \| P_2$ to be well formed the set of input channels of P_1 and P_2 must be disjoint, as must be their output channels. So, for instance, it is not possible to form the parallel composition

$$\sum_{v \in V} a!(p_v, v).0 \| a?(x).P_1 \| a?(x).P_2 \; .$$

In this way we ensure that no confusion is introduced through synchronisation.

The parallel composition of the corresponding probabilistic event structures s got by CCS parallel composition followed by restricting away events in a set S:

$$(E_1 \parallel E_2) \setminus S$$

where S consists of all labels $a!v$, $a?v$ for which $a!v$ appears in E_1 and $a?v$ in E_2, or vice versa. In this way any communication between E_1 and E_2 is forced when possible. The newly introduced τ-events, corresponding to a synchronisation between an $a!v$-event with probability p_v and an $a?v$-event with probability 1, are assigned probability p_v.

A closed conditional (**if** b **then** P_1 **else** P_2) has the denotation of P_1 when b is true and of P_2 when b is false.

The recursive definition of probabilistic event structures follows that of event structures [Win87] carrying the extra probabilities along, though care must be taken to ensure that a confusion-free event structure results.

9 Exercises

Questions (Q) and Exercises (E). Exercises can be used for evaluation, questions are more something one asks oneself.

9.1 Section 1

(Q) Why is the study of semantic models relevant?
(Q) We said that algorithms can use random choices. Human beings can flip a coin to obtain a random bit. How do computers do?
(Q) What is the difference between "probabilistic" models, and "stochastic" models. Which ones are the more general?

9.2 Section 2

(Q) What does it mean "to pick a natural number at random"? What it's the probability that this number is 42?

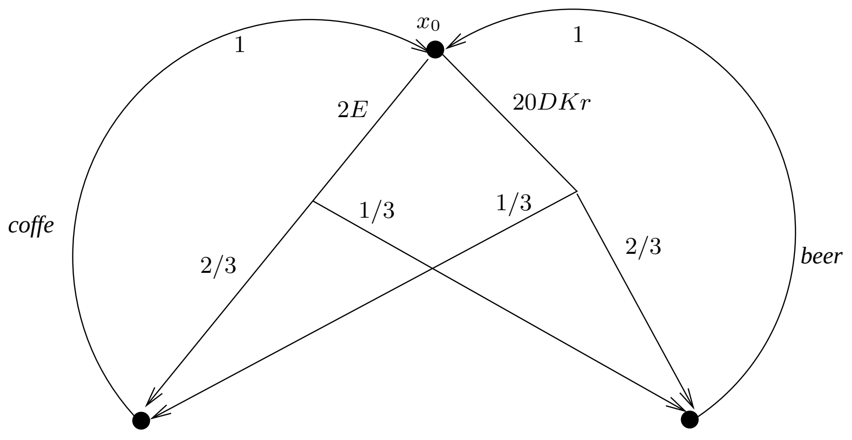
Consider the Lebesgue probability measure on $[0, 1]$.
(E) Fact: not all subsets of $[0, 1]$ are Borel. Use this fact to prove that the Borel σ-algebra is not closed under arbitrary union.
(E) What is the probability that a real number picked at random in $[0, 1]$ is rational? Does this question make sense?
(E**) What is the probability that a real number picked at random in $[0, 1]$ does not contain the digit 5 in the decimal expansion? Does this question make sense?
(E***) Show a set which is not Borel. (This will require a bibliographic research. If you come up with a solution without looking it up you are a genius. Hint: you should use the axiom of choice!)
Consider the random walk on the integers.
(E*) Prove that the set of paths that hit 0 infinitely often is measurable.
(E**) Prove that the set of paths that hit 0 at least once is measurable. What is its measure? Suppose you know it is 1. Then what is the measure of the set of paths that hit 0 infinitely often?

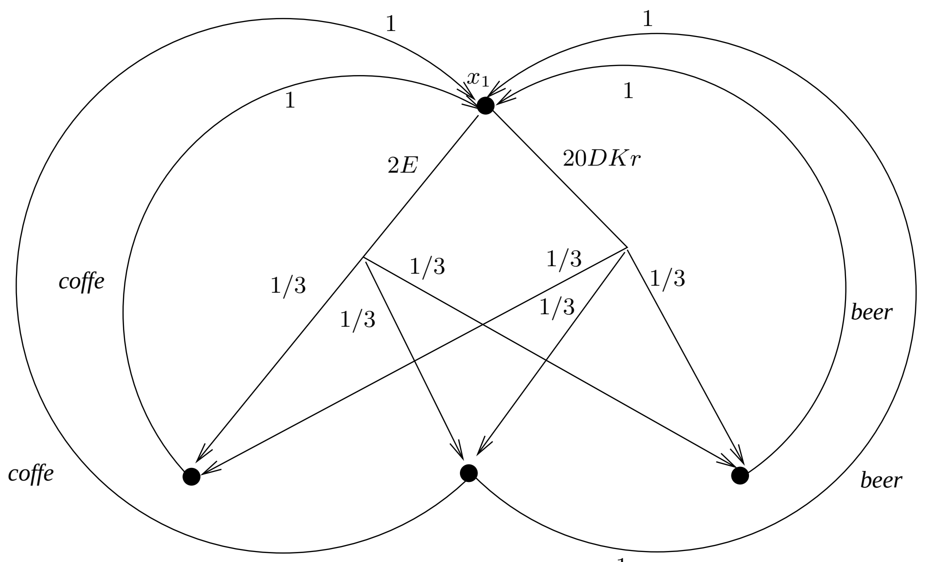
9.3 Section 3

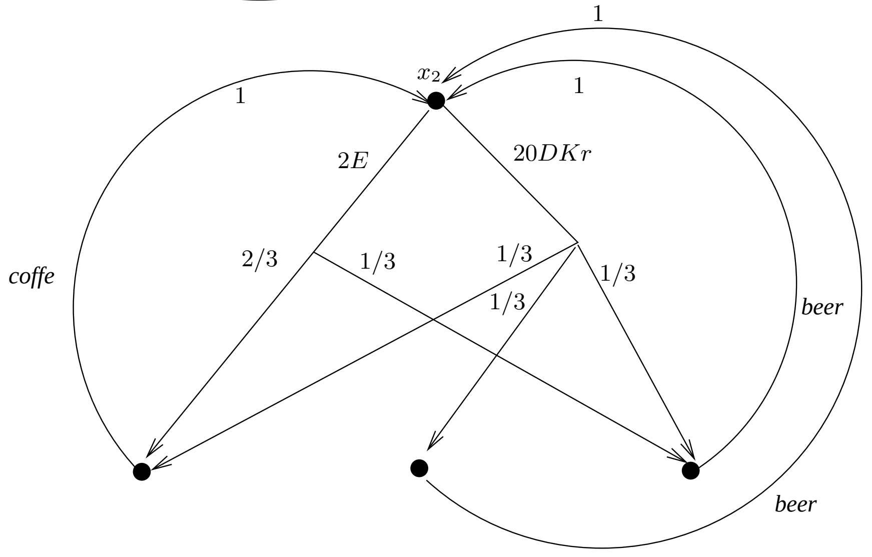
(E) In which sense is a generalised Segala automaton the most general model. Show in which way a generative LTS, a reactive LTS and a simple Segala automaton can be interpreted as generalised Segala automata.
(E) Can you interpreted a generative LTS as a simple Segala automaton?
(E) In the definitions of probabilistic LTSs we allow the use of subprobability distribution instead of just probability distribution. The missing probability represents the probability of blocking. Suppose we give an alternative definition where only probability distributions are allowed. Can you simulate subprobabilities in this restricted setting?

9.4 Section 4

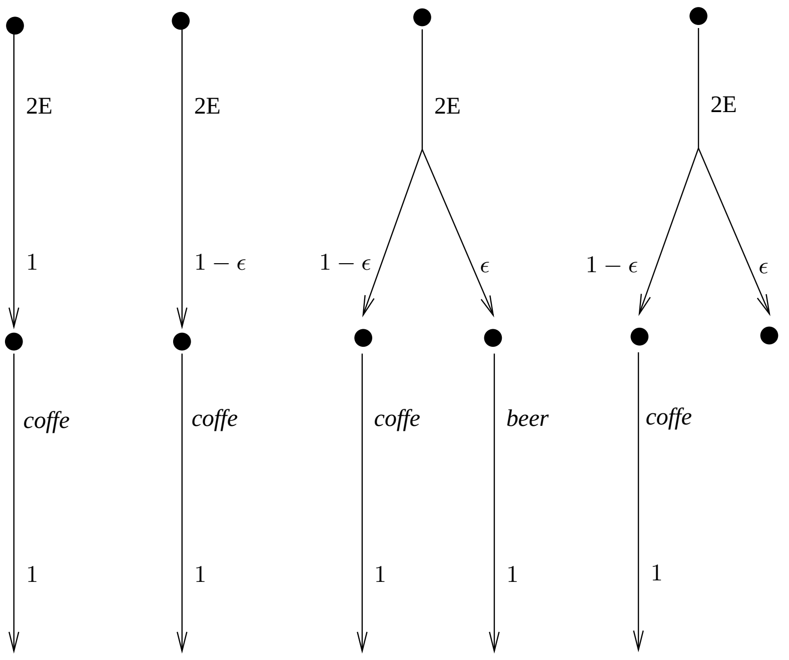
(E) If in the definition of reactive LTS we only allow probability distributions (instead of subprobability distributions), then every two processes are bisimlar. Prove it.
(E) Consider the following reactive LTSs. For every pair of systems, check whether their initial states are bisimilar. If they are, describe the bisimulation, if they are not, find a formula of the Larsen-Skou logic that distinguishes them.







(E) Consider the following reactive LTSs. For every pair of systems, find a formula of the Larsen-Skou logic without negation that distinguishes them.



(E*) Assuming the translation from Larsen-Skou logic to the functional expressions $\varphi \mapsto f$ such that

$$x \vDash \varphi \Leftrightarrow f(x) > 0$$

prove that if the distance between two states is 0, then they are bisimlar.

9.5 Section 5

(E) What is "standard rule of recursion"?
(E*) Define the semantics of the parallel composition

$$P \|_q^p Q$$

in probabilistic SCCS

9.6 Section 6

(E) Define a notion of bisimulation for Segala automata. Your definition will be OK if, when restricted to reactive LTSs it coincides with the bisimulation defined in Section 4. You may or may not need the notion of convex combination.
(E*) Define formally the notion of memoryless scheduler. Define the notion of scheduler with finite memory of size n.

9.7 Section 7

(Q) In [HP00] you find an alternative way to defining the rules COM and CLOSE. Can you think of other ways of defining it? Could we have PAR as special case? (Q) Why do we need Segala automata? Try to define a semantics using generatve or reactive LTS.

9.8 Section 8

(E*) Give a semantics to the probabilistic language of Section 8.3. Would you rather use generative LTS, reactive LTS or Segala automata?

10 Commented bibliography

For a general introduction on probability theory there are several books, see for instance [Hal50, Bil95, Wil91]. For an introduction more orientated to random processes (including Markov chains), see [GS01]. Markov decision process are also relevant to our topic, see [Put94].

As for applications of probability to computer science, many randomised distributed algorithms are discussed in [Lyn96]. An important algorithm that uses probability is the Rabin-Miller primality test [Rab80]. Probability is also very important in the context of cryptography. A good introduction is [GB99], an important classic paper is [GM84].

For probability in domain theory, the original paper [JP89] is still a gentle introduction, see also [AM00, TKP04, Var03].

One of the first probabilistic models for concurrency, the alternating model, appears in [Var85, Han91]. This is a slight variation of simple Segala automata. The reactive model was introduced in [LS91], where the notion of probabilistic bisimulation, and the Larsen-Skou logic are also introduced. The distinction between "generative" and "reactive" models goes back to [vG+90], where a semantics of a probabilistic CCS is sketched. The more refined logical characterisation of bisimulation is found in [DEP02]. A notion of metrics is defined in [D+04]. Another recent paper on the subject is [vB+03], which connects with domain theory and topology. A different notion of approximate bisimilarity appears in [DP+03].

Segala automata where introduced in [Seg95], where also a version of the logic PCTL is considered. A gentle introduction to the subject is [Sto02]. A semantic in terms of probabilistic words is proposed in [dAHJ01]. Segala automata are used for the semantics of the probabilistic π-calculus in [HP00]. A verification tool that uses Segala automata is PRISM [KNP02].

A coalgebraic presentation of all the above models, and more, is to be found in [BSdV03].

Event structures where introduced by Winskel in his PhD thesis. See the tutorial [Win87] for a good introduction. Probabilistic event structures are introduced in [VVW04].

References

[AM00] Mauricio Alvarez-Manilla. *Measure Theoretic Results for Continuous Valuations on Partially Ordered Spaces*. PhD thesis, University of London - Imperial College of Science, Technology and Medicine, 2000.

[Bil95] Patrick Billingsley. *Probability and Measure*. Wiley & Sons, New York, 1995.

[BSdV03] Falk Bartels, Ana Sokolova, and Erik de Vink. A hierarchy of probabilistic system types. In *Electronic Notes in Theoretical Computer Science*, volume 82. Elsevier, 2003.

[dAHJ01] Luca de Alfaro, Thomas A. Henzinger, and Ranjit Jhala. Compositional methods for probabilistic systems. In *Proceedings of 12th CONCUR*, volume 2154 of *LNCS*, pages 351–365. Springer, 2001.

[DEP02] Josée Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled markov processes. *Information and Computation*, 179(2):163–193, 2002.

[D+04] Josée Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics fo labelled Markov processes. *TCS*, 318(3):323–354, 2004.

[DP+03] Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Quantitative relations and approximate process equivalences. In *Proceedings of 14th CONCUR*, number 2761 of *LNCS*, pages 498–512. Springer, 2003.

[GB99] Shafi Goldwasser and Mihir Bellare. Lecture notes in cryptography. Available at http://www-cse.ucsd.edu/∼mihir/crypto-lecnotes.html, 1999.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer Sustem Sciences*, 28(2):270–299, 1984.

[GS01] Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes*. Oxford University Press, 2001.

[Hal50] Paul Halmos. *Measure Theory*. van Nostrand, 1950. New edition by Springer in 1974.

[Han91] Hans Hansson. *Time and Probability in Formal Design of Distributed systems*. PhD thesis, Uppsala University, 1991.

[HP00] Mihaela Herescu and Catuscia Palamidessi. Probabilistic asynchronous π-calculus. In *Proceedings of 3rd FoSSaCS*, volume 1784 of *LNCS*, pages 146–160. Springer, 2000.

[JP89] Claire Jones and Gordon D. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of 4th LICS*, pages 186–195, 1989.

[KNP02] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic symbolic model checker. In *Proceedings of 12th TOOLS*, volume 2324 of *LNCS*, pages 200–204. Springer, 2002. `http://www.cs.bham.uk/∼dxp/prism/`.

[LS91] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[Lyn96] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[Mil99] Robin Milner. *Communicating and Mobile Systems: The Pi Calculus*. Cambrige University Press, 1999.

[NC00] Michael Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. Cambridge, 2000.

[Put94] Martin L. Puterman. *Markov decision processes : discrete stochastic dynamic programming*. Wiley, New York, 1994.

[Rab80] Michael O. Rabin. Probabilistic algorithms for testing primality. *J. Number Theory*, 12:128–138, 1980.

[Seg95] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, M.I.T., 1995.

[Sto02] Mariëlle Stoelinga. An introduction to probabilistic automata. *Bulletin of the European Association for Theoretical Computer Science*, 78:176–198, 2002.

[TKP04] Regina Tix, Klaus Keimel, and Gordon D. Plotkin. Semantic domains for combining probability and non-determinism. Available at http://homepages.inf.ed.ac.uk/gdp/publications/, April 2004.

[Var85] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of 26th FOCS*, pages 327–338, 1985.

[Var03] Daniele Varacca. *Probability, Nondeterminism and Concurrency. Two Denotational Models for Probabilistic Computation*. PhD thesis, BRICS - Aarhus University, 2003. Available at http://www.brics.dk/∼varacca.

[vB+03] Franck van Breugel, Michael Mislove, Joel Ouaknine, and James Worrell. An intrinsic characterization of approximate probabilistic bisimilarity. In *Proceedings of 6th FOSSACS*, volume 2620 of *LNCS*, pages 200–215. Springer, 2003.

[vG+90] Rob van Glabbeek et al. Reactive, generative, and stratified models of probabilistic processes. In *Proceedings of 5th LICS*, pages 130–141, 1990.

[VVW04] Daniele Varacca, Hagen Völzer, and Glynn Winskel. Probabilistic event structures and domains. In *Proceedings of 15th CONCUR*, volume 3170 of *LNCS*, pages 481–496. Springer, 2004.

[Wil91] David Williams. *Probability with Martingales*. Cambridge University Press, 1991.

[Win87] Glynn Winskel. Event structures. In *Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course*, volume 255 of *LNCS*, pages 325–392. Springer, 1987.

[WN95] Glynn Winskel and Mogens Nielsen. Models for concurrency. In *Handbook of logic in Computer Science*, volume 4. Clarendon Press, 1995.