Web Apps & Imprecise Probabilitites[⋆]

Jorge Castro[0000−0002−1390−1313], Joaquim Gabarro[0000−0003−3771−2813], Maria Serna[0000−0001−9729−8648]

ALBCOM. CS Dept. Universitat Politècnica de Catalunya, Barcelona.
{castro, gabarro, mjserna}@cs.upc.edu

Abstract. We propose a model for the behaviour of Web apps in the unreliable WWW. Web apps are described by orchestrations. An orchestration mimics the personal use of the Web by defining the way in which Web services are invoked. The WWW is unreliable as poorly maintained Web sites are prone to fail. We model this source of unreliability trough a probabilistic approach. We assume that each site has a probability to fail. Another source of uncertainty is the traffic congestion. This can be observed as a non-deterministic behaviour induced by the variability in the response times. We model non-determinism by imprecise probabilities. We develop here an ex-ante normal to characterize the behaviour of finite orchestrations in the unreliable Web. We show the existence of a normal form under such semantics for orchestrations using asymmetric parallelism.

Keywords. Web apps, orchestrations, Orc, imprecise probabilities, normal forms.

1 Introduction

The appearance of the World Wide Web [4] deeply changed our every day life and in particular the way to interact with the world. In this paper we address the following problem: How to give an ex-ante (before execution) meaning of our interaction trough the Web. We model such interactions by means of *orchestrations*. An orchestration is the sequence and conditions in which one Web service invokes other Web services in order to realize some useful task [1]. An orchestration defines the flow control from a one-party perspective (in this case us) [17]. In general, before executing a Web app (for instance to search for flight info and get hotel reservations), we have an idea of the possible outcomes of the execution. Orchestrations are designed to address two main issues:

– Interacting trough the Web using big doses of *parallelism*. We can have several browsers (like Mozilla, Chrome or Explorer) an try to get different pieces of information at the same time.

⋆ J. Castro was partially supported by the Spanish Ministry for Economy and Competitiveness (MINECO) under grant (TIN2017- 89244-R) and the recognition 2017 SGR-856 (MACDA) from AGAUR (Generalitat de Catalunya). J. Gabarro and M. Serna were partially supported by MINECO and FEDER funds under grant GRAMM (TIN2017-86727-C2-1-R) and Generalitat de Catalunya, Agéncia de Gestió d'Ajuts Universitaris i de Recerca, under project 2017 SGR 786 (ALBCOM). M. Serna was also supported by MINECO under grant BGSMath (MDM-2014-044).

– Web is *unreliable*. Sometimes the invoked service responds but others does not. Perhaps those Web services are no longer maintained or simply they are not available at this moment. A natural way to overcome unreliability is the use of *redundancy*.

The causes of uncertainty depend deeply on the universe we are dealing with. For instance, the causes of uncertainty in economy [12,9] appear to be quite different from those on the Web. When a basic service is invoked, a site call is executed, the site can provide an answer returning some information or it can fail to (broken link). Moreover, this situation is far from being stable. Usually, based on our knowledge on the site behaviour or on external information, it is feasible to assume a *priori probability* for the broken (or silent) site event. In order to minimize the risk of calling a silent site, it is usual to issue several calls to sites providing similar information. In such a case the answer that arrives first is chosen. However, there is no a priori knowledge on which site will respond first [5] because in many cases becomes too hard to get sufficient data on the environment in order to provide precise probabilistic predictions. This lack of of precise probabilistic knowledge appears when considering an indeterministic behaviour. Following [3],we propose to model non-determinism in terms of *imprecise probabilities*.

The ex-ante characterization of an orchestration, although formulated in terms of imprecise probabilities, has an obvious practical relevance. Let us consider an orchestration P that guarantees the result *great success* with an imprecise probability greater than 1/3 and obtains the *satisfactory* result with probability greater than 3/4. Let Q be another orchestration that guarantees these same results with imprecise probabilities that are greater than, respectively, 1/4 and 4/5. Depending on our particular circumstances we can choose in a reasoned way which of the two processes, P or Q, is more convenient for our interests.

Besides proposing the uncertainty model we extend the bag semantics for orchestrations proposed in [7] to deal with daemonic indeterminism through imprecise probabilities. This allows us to generalize the previous theorem on the existence of normal forms to a more general setting. Technically, Theorem 2 shows a normal form characterization for probabilistic orchestrations and Theorem 3 extends this result to the non-deterministic case. We complement this theoretical result by developing a complete example of uncertainty analysis in our proposed model. We also sketch other possible applications of imprecise probabilities to orchestrations. In order to make the paper self-contained we start introducing the Orc language [14]. We also recall the bag semantics [7] providing meaning to Orc expressions.

2 Orchestrations in reliable environments

An orchestration is a user-defined program that utilizes services on the Web. In Orc [14] services are modelled by sites which have some predefined semantics. Typical examples of services are: an eigensolver, a search engine or a database. A *site* accepts an argument and *publishes* a result value[1]. For example, a call to a search engine, $Find(s)$,

[1] The words “publishes”,“returns” and “outputs” are used interchangeably. The terms “site” and “service” are also used interchangeably.

may publish the set of sites which *currently* offer service s. A site is *silent* if it does not publish a result. A site call can publish *at most one response*. Although a site call may have a well-defined result it may be the case that a call to the site, in an untrusted environment, fails (silence). Orc contains a number of inbuilt sites: 0 is always silent while $1(x)$ always publishes x. An orchestration which composes a number of service calls into a complex computation can be represented by an Orc expression. An orchestrator may utilize any service that is available on the grid. In this paper we deal only with *finite orchestrations* where finite means: excluding iteration and recursion. Two Orc expressions P and Q can be combined using the following operators [14,11]:

- Sequence $P > x > Q(x)$: P is evaluated and, for each value v published by P, an instance $Q(v)$ is executed. If P publishes the stream, $v_1, v_2, \ldots v_n$, then $P > x > Q(x)$ publishes some permuted stream of the outputs of the calls $Q(v_1), Q(v_2), \ldots, Q(v_n)$. When the value of x is not needed we write $P \gg Q$.
- Symmetric Parallelism $P \mid Q$: P and Q are evaluated in parallel. $P \mid Q$ publishes *some* interleaving of the streams published by P and Q.
- Asymmetric Parallelism $P(x) < x < Q$: P and Q are evaluated in parallel. Some sub-expressions in P may become blocked by a dependency on x. The first result published by Q is bound to x, the remainder of Q's evaluation is terminated and evaluation of the blocked residue of P is resumed.

Usually orchestrations assume some degree of redundancy. Following an example.

Example 1. Suppose that you need to send news to a group. Usually you prefer the BBC but it is uncertain to get a result because there is a call for strike. In such a case, you also try to get news from the CNN, $News = (BBC \mid CNN)$. To inform the group, you send the news to Alice, but at this moment you are uncertain about her capacity to get the email, therefore you send also the news to Bob, $Emails(x) = (Alice(x) \mid Bob(x))$. Consider the orchestration $eNews = Emails(x) < x < News$.

Let us describe the behaviour of $eNews$. A call to $eNews$ spans into simultaneous (parallel) calls (or threads) to $News$ and $Emails(x)$. The call to $News$ span into parallel calls to BBC and CNN. The call to $Emails(x)$ span into parallel calls to $Alice(x)$ and $Bob(x)$ At this moment the call to $eNews$ has evolved into four simultaneous threads (the programs executing the calls) corresponding to BBC, CNN, $Alice(x)$ and $Bob(x)$.

The thread corresponding to $Alice(x)$ will remain blocked until variable x takes a value. The same will happen with $Bob(x)$. Eventually (at some future time) the calls to BBC and CNN return. Assume that BBC returns first, this value will be assigned to x. Once x has a value, threads corresponding to $Alice(x)$ and $Bob(x)$ proceed and Alice and Bob will receive an email with the BBC news. Another result is possible if CNN returns first. In this case Alice and Bob get the CNN info. Note that $eNews$ has no control about which result will appear, therefore is a non-deterministic program. □

To reason about a program, we need a semantics *to assign meaning* [6]. Bag semantics was introduced in [7] to give a precise description of the approach taken in Example 1. In such approach we abstract from return time. First, let us start from the operational semantics introduced in [14]. In such a model any variable x contains all the possible values before being used. Therefore, variables keep a stream of values. When an orchestration E publishes a stream $v_1, v_2, \ldots, v_n$, *the relative ordering of the values depends on the relative response time of the sites* appearing in E. However, when we are uncertain about return times, is a strongly desirable *abstract from time*. In such a case we forget about orderings in the streams describing them as a multi-set or bag $\lfloor\lfloor v_1, v_2, \ldots, v_n \rfloor\rfloor$ (notation $\lfloor\lfloor \cdot \rfloor\rfloor$ is taken from [13]). In such a case, the "meaning" of E, denoted by $[\![E]\!]$, is the bag $\lfloor\lfloor v_1, v_2, \ldots, v_n \rfloor\rfloor$ and we write $[\![E]\!] = \lfloor\lfloor v_1, v_2, \ldots, v_n \rfloor\rfloor$. The fact that site 0 never returns is formalized as site 0 returns nothing, that is $[\![0]\!] = \lfloor\lfloor\ \rfloor\rfloor = \emptyset$. As the pruning operator (or parallel asymmetric composition) can give rise to a non deterministic behaviour, we consider also the "daemonic choice" operator $\sqcap$ [13, p. 4] to denote non deterministic choice. Toni Hoare in [10], considers the non-deterministic choice $P \sqcap Q$ between processes P and Q. In such a case $P \sqcap Q$ denotes a process which behaves like P or Q, where the selection is done without knowledge or control of the external environment. Such a choice is called *daemonic choice*. A semantic characterization of $P \sqcap Q$ in terms of *refusal sets* can be found in [10]. In a reliable environment, a call to a site S always returns a value and we write $[\![S]\!] = \lfloor\lfloor \mathtt{s} \rfloor\rfloor$.

In the following examples we justify the use of bags and how they can be obtained from the simple bags corresponding to site calls.

Example 2. $[\![BBC]\!] = \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor$ and $[\![CNN]\!] = \lfloor\lfloor \mathtt{cnn} \rfloor\rfloor$. A call to $News$ in the preceding Example 1 returns a bag containing two items, $[\![News]\!] = [\![BBC \mid CNN]\!] = \lfloor\lfloor \mathtt{bbc}, \mathtt{cnn} \rfloor\rfloor$. This result is consistent with the idea that in $News$ the BBC and the CNN are called in parallel an they return at different moments. The orchestration has no control on which one will return first. The bag $\lfloor\lfloor \mathtt{bbc}, \mathtt{cnn} \rfloor\rfloor$ mimics the idea that eventually we will get both results but we forget temporal information. □

Sometimes we want to introduce redundancy as, for example in $TwiceBBC = (BBC \mid BBC)$. Observe that this orchestration returns $\lfloor\lfloor \mathtt{bbc}, \mathtt{bbc} \rfloor\rfloor$ mimicking the idea of getting twice the same result. Showing the need of using *bags* (or *multisets*). Sometimes we get expressions that depend on the values that a variable gets during the execution. In such a case, the bag semantics provides a meaning for the variable that is used to derive the meaning of the expression. Besides in a reliable environment, asymmetric parallelism introduces indeterminism. The following example illustrates those traits.

Example 3. Now we face the meaning of x appearing in $eNews = Emails(x) < x < News$. According to Example 2 we have $[\![News]\!] = \lfloor\lfloor \mathtt{bbc}, \mathtt{cnn} \rfloor\rfloor$. As we do not control explicitly the return times, under some (external and uncontrolled) circumstances, a call $News$ returns bbc but in some other cases it returns cnn. So, x can hold either of both values, i.e., $[\![x]\!] = \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor \sqcap \lfloor\lfloor \mathtt{cnn} \rfloor\rfloor$. Assuming $[\![Emails(\mathtt{x})]\!] = \lfloor\lfloor \mathtt{alice\_x}, \mathtt{bob\_x} \rfloor\rfloor$,

$$\begin{aligned} [\![eNews]\!] &= [\![Emails(\mathtt{bbc})]\!] \sqcap [\![Emails(\mathtt{cnn})]\!] \\ &= \lfloor\lfloor \mathtt{alice\_bbc}, \mathtt{bob\_bbc} \rfloor\rfloor \sqcap \lfloor\lfloor \mathtt{alice\_cnn}, \mathtt{bob\_cnn} \rfloor\rfloor \end{aligned}$$

This result translates the idea that, depending on external circumstances, two possible output streams are possible: Alice and Bob get the BBC or Alice and Bob get the CNN.The orchestrator has no control on which one will occur. □

Working in a similar way with the different operations the existence of a normal form can be shown.

Theorem 1 ([7]). *Given an Orc expression E it holds that either $[\![E]\!] = \lfloor\lfloor\ \rfloor\rfloor$ or there is a unique non-deterministic finite decomposition in multi-sets $[\![E]\!] = \sqcap_i M_i$, where elements in M_i corresponds to the possible values returned by site calls.*

3 Orchestrations & probabilistic information

Until now, we have considered reliable orchestrations as we were certain about returns. In this section, we consider unreliable settings modelled with probabilities. Let $\Delta_n = \{(p_1, \ldots, p_n) \mid p_i \geq 0, 1 \leq i \leq n, \sum_{i=1}^{n} p_i = 1\}$. We adopt from [13] the notation $(\mathit{prog_1}@p_1 \parallel \mathit{prog_2}@p_2 \parallel \cdots \parallel \mathit{prog_n}@p_n)$ where $(p_1, \ldots, p_n) \in \Delta_n$ and $(\mathit{prog_1}, \ldots, \mathit{prog_n})$ are sequential programs to represent a probabilistic program that behaves like $\mathit{prog_i}$ with probability p_i.

The probabilistic choice follows two natural laws [15,13]. When the same program $\mathit{prog_1}$ appears twice, we should add the probabilities:

$$\begin{aligned}&(\mathit{prog_1}@p_1 \parallel \mathit{prog_1}@p_2 \parallel \mathit{prog_3}@p_3 \parallel \cdots \parallel \mathit{prog_n}@p_n)\\ &\quad = \big(\mathit{prog_1}@(p_1+p_2) \parallel \mathit{prog_3}@p_3 \parallel \cdots \parallel \mathit{prog_n}@p_n\big).\end{aligned}$$

The second rule assumes distributivity in respect to the daemonic choice operator.

$$\begin{aligned}&\big((\mathit{prog_1} \sqcap \mathit{prog'_1})@p_1 \parallel \mathit{prog_2}@p_2 \parallel \cdots \parallel \mathit{prog_n}@p_n\big)\\ &\quad = (\mathit{prog_1}@p_1 \parallel \cdots \parallel \mathit{prog_n}@p_n) \sqcap (\mathit{prog'_1}@p_1 \parallel \mathit{prog_2}@p_2 \parallel \cdots \parallel \mathit{prog_n}@p_n).\end{aligned}$$

Sometimes we can model a faulty uncertain behaviour by a probability distribution on the involved processes, but this is not always possible. We have two semantic models for faulty behaviour.

- *Probabilistic information*. In this case we model a faulty site as a site S returning $\mathtt{s}$ with a probability p, and failing to return (behaves like site 0) with probability $(1-p)$. The faulty version of S, denoted as $S_{\mathcal{F}}$ is $S_{\mathcal{F}} = (S@p \parallel 0@(1-p)))$. Moreover $[\![S_{\mathcal{F}}]\!] = \big([\![S]\!]@p \parallel \lfloor\lfloor\ \rfloor\rfloor@(1-p)\big) = \big(\lfloor\lfloor\mathtt{s}\rfloor\rfloor@p \parallel \lfloor\lfloor\ \rfloor\rfloor@(1-p)\big)$. We identify $\big(\lfloor\lfloor\mathtt{s}\rfloor\rfloor@1 \parallel \lfloor\lfloor\ \rfloor\rfloor@0\big) = \lfloor\lfloor\mathtt{s}\rfloor\rfloor$ and $\big(\lfloor\lfloor\mathtt{s}\rfloor\rfloor@0 \parallel \lfloor\lfloor\ \rfloor\rfloor@1\big) = \lfloor\lfloor\ \rfloor\rfloor$. We assume probabilistic independence on the behaviour of the sites. Two consecutive calls to a given site are considered independent in relation to its probabilistic behaviour.
- *No probabilistic information*. In such a case, we assume indeterminism, i.e., $[\![S_{\mathcal{F}}]\!] = [\![S]\!] \sqcap \lfloor\lfloor\ \rfloor\rfloor$.

When it is clear from the context that S is faulty, we denote $S_{\mathcal{F}}$ shortly as S.

Example 4. Suppose that, from a user point of view, sites CNN and BBC are unreliable, $[\![\mathit{BBC}]\!] = \big(\lfloor\lfloor\mathtt{bbc}\rfloor\rfloor@2/3 \parallel \lfloor\lfloor\ \rfloor\rfloor@1/3\big)$ and $[\![\mathit{CNN}]\!] = \big(\lfloor\lfloor\mathtt{cnn}\rfloor\rfloor@1/2 \parallel \lfloor\lfloor\ \rfloor\rfloor@1/2\big)$. A precise semantics for $[\![\mathit{News}]\!]$ comes from the way in which probabilities interact with parallel composition [7]. Assuming independence among executions, the probabilistic behaviour is given in the following table:

	$\lfloor\lfloor \texttt{bbc} \rfloor\rfloor @ 2/3$	$\lfloor\lfloor\ \rfloor\rfloor @ 1/3$
$\lfloor\lfloor \texttt{cnn} \rfloor\rfloor @ 1/2$	$\lfloor\lfloor \texttt{cnn}, \texttt{bbc} \rfloor\rfloor @ (1/2 \times 2/3)$	$\lfloor\lfloor \texttt{cnn} \rfloor\rfloor @ (1/2 \times 1/3)$
$\lfloor\lfloor\ \rfloor\rfloor @ 1/2$	$\lfloor\lfloor \texttt{bbc} \rfloor\rfloor @ (1/2 \times 2/3)$	$\lfloor\lfloor\ \rfloor\rfloor @ (1/2 \times 1/3)$

Therefore $[\![News]\!] = \left(\lfloor\lfloor \texttt{cnn}, \texttt{bbc} \rfloor\rfloor @ 1/3 \parallel \lfloor\lfloor \texttt{cnn} \rfloor\rfloor @ 1/6 \parallel \lfloor\lfloor \texttt{bbc} \rfloor\rfloor @ 1/3 \parallel \lfloor\lfloor\ \rfloor\rfloor @ 1/6\right)$. Different bags can represent the orchestration result. The empty bag $\lfloor\lfloor\ \rfloor\rfloor$ appears when both sites fail. This result is different form the one in Example 2 where only the bag $\lfloor\lfloor \texttt{cnn}, \texttt{bbc} \rfloor\rfloor$ appears. □

Probabilistic distributions are parametrized when sites are parametrized $S(x_1, \ldots x_n)$:

$$[\![S(x_1, \ldots x_n)]\!] = \begin{cases} \lfloor\lfloor s(v_1, \ldots, v_n) \rfloor\rfloor & \text{if } (x_1, \ldots x_n) = (v_1, \ldots v_n) \\ \lfloor\lfloor\ \rfloor\rfloor & \text{if } \exists i : 1 \leq i \leq n : x_i \text{ undefined} \end{cases}$$

Example 5. Suppose that *Alice* succeeds (or returns) with probability $4/5$ and *Bob* returns with probability $5/7$. In the case of *Alice* we have:

$$[\![Alice(x)]\!] = \begin{cases} \left(\lfloor\lfloor \texttt{alice\_v} \rfloor\rfloor @ 4/5 \parallel \lfloor\lfloor\ \rfloor\rfloor @ 1/5\right) & \text{if } x = v \\ \lfloor\lfloor\ \rfloor\rfloor & \text{if } x \text{ is undefined} \end{cases}$$

When $x = \lfloor\lfloor \texttt{cnn} \rfloor\rfloor$ it holds $[\![Alice(\texttt{cnn})]\!] = \left(\lfloor\lfloor \texttt{alice\_cnn} \rfloor\rfloor @ 4/5 \parallel \lfloor\lfloor\ \rfloor\rfloor @ 1/5\right)$. When x is undefined, $x = \lfloor\lfloor\ \rfloor\rfloor$ and $[\![Alice(\lfloor\lfloor\ \rfloor\rfloor)]\!] = \lfloor\lfloor\ \rfloor\rfloor$. When it is clear from the context, we write $[\![Alice(x)]\!] = \left(\lfloor\lfloor \texttt{alice\_x} \rfloor\rfloor @ 4/5 \parallel \lfloor\lfloor\ \rfloor\rfloor @ 1/5\right)$ assuming implicitly that, when x is undefined $[\![Alice(x)]\!] = \lfloor\lfloor\ \rfloor\rfloor$. Let $Emails(x)$ be $(Alice(x) \mid Bob(x))$. The semantics is

$$\left(\lfloor\lfloor \texttt{alice\_x}, \texttt{bob\_x} \rfloor\rfloor @ 4/7 \parallel \lfloor\lfloor \texttt{alice\_x} \rfloor\rfloor @ 8/35 \parallel \lfloor\lfloor \texttt{bob\_x} \rfloor\rfloor @ 1/7 \parallel \lfloor\lfloor\ \rfloor\rfloor @ 2/35\right).$$

When $x = \lfloor\lfloor \texttt{cnn} \rfloor\rfloor$, the semantics of $Emails(\texttt{cnn})$ is

$$\left(\lfloor\lfloor \texttt{alice\_cnn}, \texttt{bob\_cnn} \rfloor\rfloor @ 4/7 \parallel \lfloor\lfloor \texttt{alice\_cnn} \rfloor\rfloor @ 8/35 \parallel \lfloor\lfloor \texttt{bob\_cnn} \rfloor\rfloor @ 1/7 \parallel \lfloor\lfloor\ \rfloor\rfloor @ 2/35\right),$$

and, when x is undefined, $[\![Emails(\texttt{x})]\!] = \lfloor\lfloor\ \rfloor\rfloor$. □

The tools described in Examples 4 and 5 can be generalized. We can show that when there is no asymmetric parallelism (no indeterminism) probabilistic information can be carried out through constructors. Based on the approach given in [7], we can shown the existence of a normal form.

Theorem 2. *Let E be a finite orchestration, defined trough sequencing and parallel composition over n different faulty sites. Assume site i succeeds and returns a value with probability p_i. Under the bag semantics, there is a probabilistic finite choice decomposition in multisets $[\![E]\!] = \parallel_j M_j @ F_j(p)$, elements in M_j corresponds to the possible values returned by site calls, parameter p is the success probability vector $(p_1, \ldots, p_n)$, and F_j is an arithmetic expression defined on p.*

4 Daemonic choice & imprecise probabilities

We consider in this section the more general case of a non-deterministic orchestration defined on a faulty environment. In order to provide a semantic characterization, we keep probabilistic information as much as possible and encode non-deterministic choices as imprecise probabilities. In this way, the behaviour of a non-deterministic choice on n processes $P_1 \sqcap P_2 \sqcap \cdots \sqcap P_n$ corresponds to any of the possible behaviours defined by an imprecise probability choice:

$$\{(P_1@p_1 \parallel P_2@p_2 \parallel \cdots \parallel P_n@p_n) \mid (p_1, p_2, \ldots p_n) \in \Delta_n)\}.$$

Consequently, we identify the meaning $[\![P_1 \sqcap P_2 \sqcap \cdots \sqcap P_n]\!]$ with

$$\{([\![P_1]\!]@p_1 \parallel [\![P_2]\!]@p_2 \parallel \cdots \parallel [\![P_n]\!]@p_n) \mid (p_1, p_2, \ldots p_n) \in \Delta_n)\}.$$

Let us observe that in the asymmetric parallelism operation the non deterministic choices are restricted to the selection of an element from a multiset. The following example attempts to to grasp the relation between our approach and imprecise probabilities.

Example 6. To assign a meaning to $eNews$, recall from Example 4 that

$$[\![News]\!] = \big(\lfloor\lfloor \mathtt{cnn}, \mathtt{bbc} \rfloor\rfloor@1/3 \parallel \lfloor\lfloor \mathtt{cnn} \rfloor\rfloor@1/6 \parallel \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor@1/3 \parallel \lfloor\lfloor\ \rfloor\rfloor@1/6\big)$$

We like to keep this probabilistic information as much as possible in $[\![x]\!]$. As $[\![x]\!]$ has to be some possible multisets with at most one element we translate $\lfloor\lfloor \mathtt{cnn}, \mathtt{bbc} \rfloor\rfloor$ into $\lfloor\lfloor \mathtt{cnn} \rfloor\rfloor \sqcap \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor$. In a first approach $[\![x]\!]$ should be the process

$$\Big(\big(\lfloor\lfloor \mathtt{cnn} \rfloor\rfloor \sqcap \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor\big)@1/3 \parallel \lfloor\lfloor \mathtt{cnn} \rfloor\rfloor@1/6 \parallel \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor@1/3 \parallel \lfloor\lfloor\ \rfloor\rfloor@1/6\Big).$$

Modelling non-determinism by imprecise probabilities

$$[\![\big(\lfloor\lfloor \mathtt{cnn} \rfloor\rfloor \sqcap \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor\big)]\!] = \{\lfloor\lfloor \mathtt{cnn} \rfloor\rfloor@p_1 \parallel \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor@p_2 \mid (p_1, p_2) \in \Delta_2\}$$

Substituting in the previous expression, we get

$$\begin{aligned}[\![x]\!] \;&= \Big\{\big(\lfloor\lfloor \mathtt{cnn} \rfloor\rfloor@\frac{1}{2}p_1 \parallel \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor@\frac{1}{3}p_1 \parallel \lfloor\lfloor\ \rfloor\rfloor@\frac{1}{6}p_1 \\ &\qquad \parallel \lfloor\lfloor \mathtt{cnn} \rfloor\rfloor@\frac{1}{6}p_2 \parallel \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor@\frac{2}{3}p_2 \parallel \lfloor\lfloor\ \rfloor\rfloor@\frac{1}{6}p_2\big) \mid (p_1, p_2) \in \Delta_2\Big\}.\end{aligned}$$

Regrouping terms we get for $[\![x]\!]$

$$\Big\{\Big(\lfloor\lfloor \mathtt{cnn} \rfloor\rfloor@\Big(\frac{1}{2}p_1 + \frac{1}{6}p_2\Big) \parallel \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor@\Big(\frac{1}{3}p_1 + \frac{2}{3}p_2\Big) \parallel \lfloor\lfloor\ \rfloor\rfloor@\frac{1}{6}\Big) \mid (p_1, p_2) \in \Delta_2\Big\}$$

Observe that, although the characterization of $[\![x]\!]$ is imprecise, we can assure that the probability of having cnn as a result lies in the interval $[1/6, 1/2]$ and that the probability of bbc lies in $[1/3, 2/3]$. We rewrite $[\![x]\!]$ as :

$$\{(\lfloor\lfloor \mathtt{cnn} \rfloor\rfloor@p \parallel \lfloor\lfloor \mathtt{bbc} \rfloor\rfloor@q \parallel \lfloor\lfloor\ \rfloor\rfloor@1/6) \mid p \in [1/6, 1/2], q \in [1/3, 2/3], p + q = 5/6\}$$

Assuming the semantics of $Emails(x)$ in Example 5, then $[\![eNews]\!]$ is

$$\Big\{\big(\ \|\ \lfloor\lfloor \texttt{alice\_cnn}, \texttt{bob\_cnn}\rfloor\rfloor @\frac{4}{7}p \ \|\ \lfloor\lfloor \texttt{alice\_cnn}\rfloor\rfloor @\frac{8}{35}p \ \|\ \lfloor\lfloor \texttt{bob\_cnn}\rfloor\rfloor @\frac{1}{7}p$$
$$\|\ \lfloor\lfloor \texttt{alice\_bbc}, \texttt{bob\_bbc}\rfloor\rfloor @\frac{4}{7}q \ \|\ \lfloor\lfloor \texttt{alice\_bbc}\rfloor\rfloor @\frac{8}{35}q \ \|\ \lfloor\lfloor \texttt{bob\_bbc}\rfloor\rfloor @\frac{1}{7}q$$
$$\|\ \lfloor\lfloor\ \rfloor\rfloor @\frac{3}{14}\big) \mid p \in [1/6, 1/2], q \in [1/3, 2/3], p + q = 5/6\Big\}.$$

The meaning $[\![eNews]\!]$ provides, for each possible output stream, a probability interval. This quantitative information may be relevant in any discussion about the appropriateness of this orchestration. □

Our next result provides a generalization of a similar result in [7]. We are able to include asymmetric parallelism in the bag semantics and devise a normal form. The proof is by induction on the structure of the orchestration and uses formalizations of the preceding ideas.

Theorem 3. *Let E be a finite faulty orchestration, defined trough sequencing, parallel composition and asymmetric parallelism over n different faulty sites. Assume that site i succeds an returns a value with probability p_i and let $p = (p_1, \dots, p_n)$. Under the bag semantics, encoding the daemonic choice due to asymmetric parallelism into imprecise probabilities, there are multisets $M_1, \dots M_\ell$ and a Cartesian product of probability spaces $\Delta_{m_1} \times \cdots \times \Delta_{m_k}$ such that*

$$[\![E]\!] = \big\{\big(M_1 @ F_1(p, \delta) \ \|\ \cdots \ \|\ M_l @ F_l(p, \delta)\big) \mid \delta \in \Delta_{n_1} \times \cdots \times \Delta_{n_k}\big\}.$$

Multiset's elements correspond to possible values returned by by site calls and formulas F_j are arithmetic expressions defined on the success probability vector p and a tuple of distributions δ.

5 An example of application

In order to clarify the measurable probabilities $p = (p_1, \dots, p_n)$ and the source for imprecise probabilities $\delta \in \Delta_{n_1} \times \cdots \times \Delta_{n_k}$ appearing in Theorem 3 we analyse the meaning on a longer orchestration.

Consider the orchestration $EmailFlightHotel$ sending to $Alice$ information about flights and hotels. Three hotels are asked to give information: $Hotels = (H_1 \mid H_2 \mid H_3)$ and two flight companies are contacted: $Flights = (F_1 \mid F_2)$. Thus,

$$EmailFlightHotel = Alice(f, h) < f < Flights < h < Hotels$$

The set of sites is $\{H_1, H_2, H_3, F_1, F_2, Alice(f, h)\}$. The success probabilities are respectively $\{1/2, 1/3, 1/4, 1/5, 1/6, 2/3\}$. Observe that

$$\begin{aligned}[\![Hotels]\!] = \big(&\lfloor\lfloor h_1, h_2, h_3\rfloor\rfloor @1/24 \\ &\|\ \lfloor\lfloor h_1, h_2\rfloor\rfloor @3/24 \ \|\ \lfloor\lfloor h_1, h_3\rfloor\rfloor @2/24 \ \|\ \lfloor\lfloor h_2, h_3\rfloor\rfloor @1/24 \ \| \\ &\|\ \lfloor\lfloor h_1\rfloor\rfloor @6/24 \ \|\ \lfloor\lfloor h_2\rfloor\rfloor @3/24 \ \|\ \lfloor\lfloor h_3\rfloor\rfloor @2/24 \\ &\|\ \lfloor\lfloor\ \rfloor\rfloor @6/24\big)\end{aligned}$$

Note that any probability appearing in $[\![Hotels]\!]$ is a funcion of the success probability of H_1, H_2, H_3 given by $p = (1/2, 1/3, 1/4)$. For instance the $6/24$ appearing in $\lfloor\!\lfloor h_1 \rfloor\!\rfloor @ 6/24$ is computed as $6/24 = 1/2(1-1/3)(1-1/4)$. To assign a meaning to h, each multiset gets an imprecise probability on its elements. That is

$$\begin{aligned}
\lfloor\!\lfloor h_1, h_2, h_3 \rfloor\!\rfloor &= \big\{\big(\lfloor\!\lfloor h_1 \rfloor\!\rfloor @ p_{1,1} \parallel \lfloor\!\lfloor h_2 \rfloor\!\rfloor @ p_{1,2} \parallel \lfloor\!\lfloor h_3 \rfloor\!\rfloor @ p_{1,3}\big) \mid (p_{1,1}, p_{1,2}, p_{1,3}) \in \Delta_3\big\} \\
\lfloor\!\lfloor h_1, h_2 \rfloor\!\rfloor &= \big\{\big(\lfloor\!\lfloor h_1 \rfloor\!\rfloor @ p_{2,1} \parallel \lfloor\!\lfloor h_2 \rfloor\!\rfloor @ p_{2,2} \parallel \big) \mid (p_{2,1}, p_{2,2}) \in \Delta_2\big\} \\
\lfloor\!\lfloor h_1, h_3 \rfloor\!\rfloor &= \big\{\big(\lfloor\!\lfloor h_1 \rfloor\!\rfloor @ p_{3,1} \parallel \lfloor\!\lfloor h_3 \rfloor\!\rfloor @ p_{3,3}\big) \mid (p_{3,1}, p_{3,3}) \in \Delta_2\big\} \\
\lfloor\!\lfloor h_2, h_3 \rfloor\!\rfloor &= \big\{\big(\lfloor\!\lfloor h_2 \rfloor\!\rfloor @ p_{4,2} \parallel \lfloor\!\lfloor h_3 \rfloor\!\rfloor @ p_{4,3}\big) \mid (p_{4,2}, p_{4,3}) \in \Delta_2\big\}
\end{aligned}$$

Define $\delta = (p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}, p_{3,1}, p_{3,3}, p_{4,2}, p_{4,3}) \in \Delta_3 \times \Delta_2 \times \Delta_2 \times \Delta_2$, then: $[\![h]\!] = \big\{\big(\lfloor\!\lfloor h_1 \rfloor\!\rfloor @ P_{h_1} \parallel \lfloor\!\lfloor h_2 \rfloor\!\rfloor @ P_{h_2} \parallel \lfloor\!\lfloor h_3 \rfloor\!\rfloor @ P_{h_3} \parallel \lfloor\!\lfloor\ \rfloor\!\rfloor @ P_{h_\emptyset} \parallel \big)\big\}$ where:

$$\begin{aligned}
P_{h_1} &= \frac{1}{24}(p_{1,1} + 3p_{2,1} + 2p_{3,1} + 6) & P_{h_2} &= \frac{1}{24}(p_{1,2} + 3p_{2,2} + p_{4,2} + 3) \\
P_{h_3} &= \frac{1}{24}(p_{1,3} + 2p_{3,3} + p_{4,3} + 2) & P_{h_\emptyset} &= \frac{6}{24}
\end{aligned}$$

Note that P_{h_i} is a function of p and δ previously defined, that is $P_{h_i} = F_{h_i}(p, \delta)$, similarly for $P_{h_\emptyset}$.

Working in a similar way.

$$[\![Flights]\!] = \big(\lfloor\!\lfloor f_1, f_2 \rfloor\!\rfloor @ 1/30 \parallel \lfloor\!\lfloor f_1 \rfloor\!\rfloor @ 5/30 \parallel \lfloor\!\lfloor f_2 \rfloor\!\rfloor @ 4/30 \parallel \lfloor\!\lfloor\ \rfloor\!\rfloor @ 20/30\big)$$

In this case the probabilities appearing in $[\![Flights]\!]$ are function of $p' = (1/5, 1/6)$. There is just one bag with more than one element, then

$$\lfloor\!\lfloor f_1, f_2 \rfloor\!\rfloor = \{\big(\lfloor\!\lfloor f_1 \rfloor\!\rfloor @ q_{1,1} \parallel \lfloor\!\lfloor f_2 \rfloor\!\rfloor @ q_{1,2} \mid (q_{1,1}, q_{1,2}) \in \Delta_2\big)\}$$

Defining $\delta' = (q_{1,1}, q_{1,2}) \in \Delta_2$. So, $[\![f]\!] = \{(\lfloor\!\lfloor f_1 \rfloor\!\rfloor @ Q_{f_1} \parallel \lfloor\!\lfloor f_2 \rfloor\!\rfloor @ Q_{f_2} \parallel \lfloor\!\lfloor\ \rfloor\!\rfloor @ Q_{f_\emptyset})\}$ where

$$Q_{f_1} = \frac{1}{30}(q_{1,1} + 5),\ Q_{f_2} = \frac{1}{30}(q_{1,2} + 4),\ Q_{f_\emptyset} = \frac{20}{30}.$$

Finally,

$$\begin{aligned}
&[\![EmailFightHotel]\!] \\
&\quad = \{\big(\lfloor\!\lfloor \texttt{alice\_f}_1\texttt{\_h}_1 \rfloor\!\rfloor @ P_1 \parallel \cdots \parallel \lfloor\!\lfloor\ \rfloor\!\rfloor @ P_\emptyset\big) \mid \cdots\} \\
&\quad = \{\big(\lfloor\!\lfloor \texttt{alice\_f}_1\texttt{\_h}_1 \rfloor\!\rfloor @ \frac{2}{3} Q_{f_1} P_{h_1} \parallel \cdots \parallel \lfloor\!\lfloor\ \rfloor\!\rfloor @ \frac{1}{3} P_{h_\emptyset} P_{f_\emptyset}\big) \mid \cdots\} \\
&\quad = \{\big(\lfloor\!\lfloor \texttt{alice\_f}_1\texttt{\_h}_1 \rfloor\!\rfloor @ \frac{2}{3}\big(\frac{1}{30} q_{1,1} + \frac{5}{30}\big)\big(\frac{1}{24} p_{1,1} + \frac{3}{24} p_{2,1} + \frac{2}{24} p_{3,1} + \frac{6}{24}\big) \parallel \cdots \\
&\qquad\quad \parallel \lfloor\!\lfloor\ \rfloor\!\rfloor @ \frac{1}{3} \cdot \frac{6}{24} \cdot \frac{20}{30}\big) \mid \cdots\}
\end{aligned}$$

Define p'' as the array associated to the $Alice$ probability of success, $p'' = (2/3)$. Defining (with a small abuse of notation):

$$\begin{aligned}
\delta &= (\delta, \delta') = (p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}, p_{3,1}, p_{3,3}, p_{4,2}, p_{4,3}, q_{1,1}, q_{1,2}) \\
p &= (p, p', p'') = (1/2, 1/3, 1/4, 1/5, 1/6, 2/3)
\end{aligned}$$

We have got P_1 as an arithmetic expression on (p, δ) like the probability expressions in Theorem 3. Other cases are similar.

6 Other applications

We consider briefly two settings to which we can extend the preceding approach. We started from the case of fully reliable sites to include probabilistic (but reliable) sites. We can consider the case where sites are fully reliable (response is granted) with uncertain response time. As before we encode demonic choice as an imprecise probability. This case, although a special case of the preceding one, merits special attention because the empty bag cannot appear. Even if the result is uncertain it is less uncertain than in the faulty case. Let us develop those ideas through an example.

Example 7. Let us consider $eNews$ introduced in Example 1. According to Example 3 we have $[\![x]\!] = \lfloor\!\lfloor \texttt{bbc} \rfloor\!\rfloor \sqcap \lfloor\!\lfloor \texttt{cnn} \rfloor\!\rfloor$. As sites always return, the only probabilities are due to the indeterministic choice and therefore $[\![x]\!]$ is less ambiguous that in Example 6,

$$[\![x]\!] = \{(\lfloor\!\lfloor \texttt{bbc} \rfloor\!\rfloor @ p_1 \parallel \lfloor\!\lfloor \texttt{cnn} \rfloor\!\rfloor @ p_2) \mid (p_1, p_2) \in \Delta_2)\}$$

Then $[\![eNews]\!] = \{([\![Emails(\texttt{bbc})]\!] @ p_1 \parallel [\![Emails(\texttt{cnn})]\!] @ p_2) \mid (p_1, p_2) \in \Delta_2\}$. The main difference between this example and Example 6 is that both, Alice and Bob, will receive one newspaper for sure but is not known which one. Note that in Example 6 both (Alice and Bob) just one (Alice or Bob) or no-one (neither Alice nor Bob) get a newspaper. □

We have used probabilistic information only on site failures. However, our approach can be extended to orchestrations having probabilistic behaviour. The following examples provides the main ideas in this setting.

Example 8. We assume that all the sites have a reliable behaviour. Consider a probabilistic site (modelled by an orchestration) $infoNews = (BBC@3/4 \parallel CNN@1/4)$ returning news form the BBC with probability 3/4 or CNN with probability 1/4. Note that $infoNews$ returns a result with probability 1, that is $[\![infoNews]\!] \neq \lfloor\!\lfloor\ \rfloor\!\rfloor$. Based on the previous site define $otherNews = (infoNews \mid DISNEY)$ and finally:

$$pr\_toAlice = Alice(x) < x < otherNews.$$

Clearly $[\![otherNews]\!] = (\lfloor\!\lfloor \texttt{bbc}, \texttt{disney} \rfloor\!\rfloor @ 3/4 \parallel \lfloor\!\lfloor \texttt{cnn}, \texttt{disney} \rfloor\!\rfloor @ 1/4)$. Using indeterminism to split bags into individual responses we get

$$\begin{aligned}[\![x]\!] &= \big((\lfloor\!\lfloor \texttt{bbc} \rfloor\!\rfloor \sqcap \lfloor\!\lfloor \texttt{disney} \rfloor\!\rfloor) @ 3/4 \parallel (\lfloor\!\lfloor \texttt{cnn} \rfloor\!\rfloor \sqcap \lfloor\!\lfloor \texttt{disney} \rfloor\!\rfloor) @ 1/4\big) \\ &= (\lfloor\!\lfloor \texttt{bbc} \rfloor\!\rfloor @ 3/4 \parallel \lfloor\!\lfloor \texttt{cnn} \rfloor\!\rfloor @ 1/4) \sqcap (\lfloor\!\lfloor \texttt{bbc} \rfloor\!\rfloor @ 3/4 \parallel \lfloor\!\lfloor \texttt{disney} \rfloor\!\rfloor @ 1/4) \\ &\quad \sqcap (\lfloor\!\lfloor \texttt{cnn} \rfloor\!\rfloor @ 1/4 \parallel \lfloor\!\lfloor \texttt{disney} \rfloor\!\rfloor) @ 3/4) \sqcap \lfloor\!\lfloor \texttt{disney} \rfloor\!\rfloor.\end{aligned}$$

Translating indeterminism into imprecise probabilities, we get

$$\begin{aligned}[\![x]\!] = \Big\{ & \big(\lfloor\!\lfloor \texttt{bbc} \rfloor\!\rfloor @ \frac{3}{4}(p_1 + p_2)\big) \parallel \lfloor\!\lfloor \texttt{cnn} \rfloor\!\rfloor @ \frac{1}{4}(p_1 + p_3) \\ & \parallel \lfloor\!\lfloor \texttt{disney} \rfloor\!\rfloor @ \big(p_4 + \frac{1}{4}p_2 + \frac{3}{4}p_3\big) \mid (p_1, p_2, p_3, p_4) \in \Delta_4\}.\end{aligned}$$

Finally,

$$\llbracket pr\_toAlice \rrbracket = \Big\{ \big(\lfloor\lfloor \texttt{alice\_bbc} \rfloor\rfloor @ \frac{3}{4}(p_1 + p_2) \parallel \lfloor\lfloor \texttt{alice\_cnn} \rfloor\rfloor @ \frac{1}{4}(p_1 + p_3) \parallel \lfloor\lfloor \texttt{alice\_disney} \rfloor\rfloor @ \big(p_4 + \frac{1}{4}p_2 + \frac{3}{4}p_3\big) \mid (p_1, p_2, p_3, p_4) \in \varDelta_4 \Big\}.$$

Thus, it is granted that Alice gets a result but the type of the result is uncertain. □

7 Conclusion and open problems

The economist Frank Knight has made a distinction between risk and uncertainty [12] as illustrated by the following quotation taken from [[2], Chapter 11]:

> Risk refers to something that can be measured by mathematical probabilities. In contrast, uncertainty refers to something that cannot be measured (using probabilities) because there are no objective standards to express these probabilities.

In this paper we model the uncertainty issued by the daemonic choice, represented by $P \sqcap Q$ by the set of imprecise probabilities $\{(P@p_1 \parallel Q@p_2) \mid p_1 + p_2 = 1\}$. Imprecise probabilities overcome the Knightian problem of the existence of a unique probability. In particular we apply this approach to model the uncertain Web. Nevertheless, in asymmetric parallelism, non determinism is limited to the selection of an element from a multiset. It will be of interest to analyse, in the general context of processes' algebra, the existence of normal forms by modelling non determinism by imprecise probabilities.

In this paper we have assumed sites with well defined return probabilities. It could be also possible to consider sites with imprecise return probabilities. For instance, let CNN a site with uncertain return probability in between $[1/6, 1/2]$, then

$$\llbracket CNN \rrbracket = \{ \lfloor\lfloor \texttt{cnn} \rfloor\rfloor @p \parallel \lfloor\lfloor\ \rfloor\rfloor @(1-p) \mid p \in [1/6, 1/2] \}$$

It seems possible to extend the normal forms to this case.

In [8] another approach was undertaken to model Web uncertainty. It is assumed that sites can fail but the number of failures is bounded. As the failing sites are not known, some working hypothesis should be done between the best and the worst scenarios. It is assumed that some sites will fail trying to damage the orchestrations as much as possible (daemons $\mathfrak{d}$) but others will fail trying to minimize damage (angels $\mathfrak{a}$). This approach give rise to a strategic situation analysed trough a zero-sum game (called the $\mathfrak{a}$-$\mathfrak{d}$ game) [16]. It is an open topic if there is any relation between $\mathfrak{a}$-$\mathfrak{d}$ approach and imprecise probabilities approach.

A fundamental question in program design is: when a program is better than another? Partial orders have been considered to tackle this question. Expression $P \sqsubset Q$ points out that program Q is better than program P [10][13]. On highly unreliable environments this question is even more crucial. Although there exists a general approach [13], the application to the Web environment remains open.

Finally, in Theorem 3 a Cartesian product of probability spaces is considered. However there are situations where a richer correlation structure is suitable, or where additional information could be incorporated (i.e. more complex constraints on the probabilities directly). For instance, consider the case of locally congested network evolving along the time. These cases seems hard to study is this framework.

References

1. W3c, web services glossary, `http://www.w3.org/TR/ws-gloss/`
2. Akerlof, G., Schiller, R.: Animal Spirits. Princeton Univ. Press, Princeton and Oxford (2009)
3. Augustin, T., Coolen, F., Cooman, G., Troffaes, M.: Introduction to Imprecise Probabilities. Wiley (2014), `https://doi.org/10.1002/9781118763117`
4. Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H.F., Secret, A.: The world-wide web. Commun. ACM 37(8), 76–82 (Aug 1994), `http://doi.acm.org/10.1145/179606.179671`
5. Dean, J., Barroso, L.: The tail at scale. Commun. ACM 56(2), 74–80 (Feb 2013), `http://doi.acm.org/10.1145/2408776.2408794`
6. Floyd, R.W.: Assigning meanings to programs. In: Schwartz, J.T. (ed.) Proceedings of Symposium on Applied Mathematical Aspects of Computer Science. pp. 19–32. American Mathematical Society (1967)
7. Gabarro, J., Leon-Gaixas, S., Serna, M.: The computational complexity of QoS measures for orchestrations. J. Comb. Optim. 34(4), 1265–1301 (2017), `https://doi.org/10.1007/s10878-017-0146-9`
8. Gabarro, J., Serna, M., Stewart, A.: Analysing web-orchestrations under stress using uncertainty profiles. The Computer Journal 57(11), 1591–1615 (2014), `https://doi.org/10.1093/comjnl/bxt063`
9. Galbraith, J.K.: The Age of Uncertainty. houghhton Miffin Company, Boston (1977)
10. Hoare, C.: Communicating Sequential Processes. Prentice-Hall, London (1985)
11. Kitchin, D., Quark, A., Cook, W., Misra, J.: The Orc programming language. In: Lee, D., Lopes, A., A.Poetzsch-Heffter (eds.) Proceedings of FMOODS/FORTE 2009. Lecture Notes in Computer Science, vol. 5522. Springer (2009), `https://doi.org/10.1007/978-3-642-02138-1_1`
12. Knight, F.: Risk, uncertainty and Profit. Houghton Mifflin, Boston and New York (1921), `http://www.econlib.org/library/Knight/knRUP.html`
13. McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Springer, New York (2005)
14. Misra, J., Cook, W.: Computation orchestration: A basis for wide-area computing. Software and Systems Modeling 6(1), 83–110 (2007), `https://doi.org/10.1007/s10270-006-0012-1`
15. Morgan, C., Mclver, A., Sanders, J.W.: In: J. W. Davies, A.R., Woodcock, J. (eds.) Millennial Perspectives in Computer Science, chap. Probably Hoare? Hoare probably!, pp. 271–282. Palgrave, Basingstoke (2000)
16. von Neumann, J., Morgenstern, O.: Theory of games and economic behavior, 60th Anniversary Commemorative Edition. Princeton University Press, Princeton and Oxford (1953)
17. Peltz, C.: Web services orchestration and choreography. IEEE Computer 36(10), 46–52 (2003), `https://doi.org/10.1109/MC.2003.1236471`