Back to the format

Citation for published version (APA):

Document license:
CC BY

DOI:
10.1016/j.jlamp.2023.100929

Document status and date:
Published: 01/02/2024

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 27. Mar. 2024
Back to the format: A survey on SOS for probabilistic processes

Valentina Castiglioni a,*, Ruggero Lanotte b, Simone Tini b

a Eindhoven University of Technology, De Zaale, 5612 AZ, Eindhoven, the Netherlands
b University of Insubria, via Valleggio 11, 22100 Como, Italy

ABSTRACT

In probabilistic process algebras the classic qualitative description of process behaviour is enriched with quantitative information on it, usually modelled in terms of probabilistic weights and/or distributions over the qualitative behaviour. In this setting, we use behavioural equivalences to check whether two processes show exactly the same behaviour, and, if this is not the case, we can use behavioural metrics to measure the distance between them. Compositional reasoning requires that equivalence, or closeness, of behaviour of two processes are not destroyed when language operators are applied on top of them in order to build larger processes. Formally, the equivalence must be a congruence, and the metric must be uniformly continuous, with respect to language operators. Instead of verifying these compositional properties by hand, operator-by-operator, it is much more convenient to prove them for a class of operators once for all, and to check that the operators one is dealing with are in that class. This is achieved by means of SOS specification formats: they consist in a set of syntactical constraints characterising a class of operators on the patterns of SOS rules, that define the operational semantics of languages. With this survey, we aim to collect and describe the specification formats that have been proposed in the literature to guarantee the compositional properties of (variants of) bisimulation equivalences and bisimulation metrics in the probabilistic setting.

1. Introduction

The Structural Operational Semantics (SOS) framework was introduced by Plotkin [100] to provide a model for process algebras [20], and, thus, to equip processes with an operational semantics, i.e., a way to describe their computation steps and the (possible) state-changes in a step-by-step fashion. Briefly, the operational semantics of processes is defined as a Labelled Transition System (LTS) [83], i.e. the set of transitions (each modelling a computation step) that is generated from a collection of operational rules. Those rules allow us to infer transitions from transitions, according to the syntactic structure of processes.

Later on, following the findings of de Simone in his PhD thesis [105], the SOS framework has been successfully applied as a formal tool to devise a meta-theory of process algebras, i.e., to develop results that hold for classes of (process description, specification) languages. The most prominent example in this direction is the use of the SOS framework to favour compositional reasoning. Informally, we expect two processes that show the same behaviour to be inter-replaceable: if a system is specified as $f(p,q)$, i.e., the composition of processes $p,q$ via the language operator $f$, and $p$ shows the same behaviour of process $r$, then the behaviour of $f(r,q)$, i.e., the

* This work has been partially supported by the project Programs in the wild: uncertainties, adaptability and verification (ULTRON) of the Icelandic Research Fund (grant No. 228376-051).
* Corresponding author.

E-mail addresses: v.castiglioni@tue.nl (V. Castiglioni), ruggero.lanotte@uninsubria.it (R. Lanotte), simone.tini@uninsubria.it (S. Tini).
system in which we replace \( p \) with \( r \), should be equal to that of \( f(p,q) \). In the process algebraic setting, to formalise whether two processes show the same behaviour we make use of behavioural equivalences. These are equivalence relations defined over LTSs that allow us to establish whether the behaviours of two processes are indistinguishable for their observers. In the literature, several notions of behavioural equivalences have been proposed, each based on how much we abstract from the information carried by the LTS: for instance, a classic behavioural equivalence is (strong) bisimilarity [95,100], which identifies two processes if they can mimic each other’s transitions and reach processes that are, in turn, equivalent. The interested reader is referred to the taxonomies in [69,71] for an exhaustive presentation of behavioural equivalences. A behavioural equivalence that is compatible with all language operators, i.e., for which the property of inter-replacement outlined above holds with respect to any operator \( f \) of the language, is called a congruence.

Given a behavioural equivalence and a language, one could prove the desired compositional property, like that of congruence, in a direct way, operator-by-operator. However, this approach is time consuming, and typically ends up in the development of long technical results that hold specifically over the considered language and, thus, cannot be reused to verify the same property on a different language. Since compositional results can be established from general properties of operators, it is much more convenient to identify the class of operators for which the desired property holds, and then check whether the operators of the considered language belong to that class. Rule formats were introduced with this exact purpose: they are a set of syntactical constraints over the operational rules ensuring that if all the rules for the language satisfy them, then the desired property holds over the generated LTS. Rule formats become specification formats if also some constraints on the set of rules are imposed. In particular, rule and specification formats designed to guarantee the congruence property are also called congruence formats.

In the last three decades, a variety of studies on the compositional properties of processes have been conducted. We refer the interested reader to the survey papers [1,98] for a presentation of the rule formats proposed in the classic process algebras literature. However, the advances made in the last decade on the modelling and behavioural analysis of probabilistic processes have called for new results on compositional reasoning. This survey is then a natural follow-up to [1,98] that focuses on probabilistic process algebras.

1.1. This survey: formats and probability

In probabilistic process algebras, the classic qualitative description of process behaviour outlined above is enriched with quantitative, probabilistic, information on it. In particular, the transitions in an LTS have to be modified in order to take also the probabilistic behaviour into account. However, following various interpretations of the interplay of nondeterminism and probability, as well as different perspectives on the probabilistic behaviour, in the literature we can find a wealth of proposals of probabilistic extensions of LTSs. To the best of our knowledge, there are only three probabilistic models for which specification formats have been proposed: the generative model (GPTS) [12,15,29], the reactive model (RPTS) [72,88], and the nondeterministic and probabilistic model (NPTS) [102]. Briefly, in GPTSs nondeterminism is replaced by probabilistic choices; in RPTSs the possible reactions of a process to an input from the external environment are determined probabilistically; in NPTSs each nondeterministic choice by a processes induces a probability distribution over the next-state behaviour. Clearly, each model comes with its own syntactic presentation, and, thus, with its own formats. Moreover, the same wealth of approaches can be found, even enriched, in the definition of behavioural equivalences over each model, where the choice on how probabilities have to be taken into account and compared plays a fundamental role. We refer the interested reader to [22,24,77] for presentations of the different notions of probabilistic equivalences. A common feature to behavioural equivalences in the classical (i.e., non-probabilistic) and in the probabilistic settings, is that they relate processes that behave exactly the same. However, the information on probabilistic behaviour usually derives from statistical samplings, or measures on physical systems, and, thus, they are inevitably subject to errors and approximations. Consequently, one can be interested to know whether the behaviour of two processes is similar up-to some tolerance, or to evaluate how far the behaviour of two processes is apart. This led to the introduction of approximated equivalences (see, e.g., [3,52,53,67]) and behavioural metrics (see, e.g., [30,35,43,50,86]). The former ones quantify the differences in the probabilities to perform a single computation step. Conversely, the latter ones measure the differences arising along the entire computation.

Regarding compositional reasoning, what means to be compatible with language operators now depends also on whether we are considering (approximated) equivalences or metrics. While for probabilistic equivalences, this compatibility is still expressed in terms of congruence properties, in the case of approximated relations and metrics we need to ensure that the induced distances satisfy some uniform continuity properties with respect to the language operators. Informally, a small variation in the behaviour of processes should cause only a bounded variation in the behaviour of their composition: if process \( p \) differs by at most \( \epsilon \) from process \( r \), then the difference between \( f(p,q) \) and \( f(r,q) \) should be bounded by \( m(\epsilon) \), for a proper function \( m \), called modulus of continuity, that can correspond to non-extensiveness [9], non-expansiveness [51], or Lipschitz continuity [62].

In this survey we collect and discuss the formats for probabilistic behavioural equivalences, approximated equivalences, and behavioural metrics, over GPTSs, RPTSs, and/or NPTSs, that have been proposed in the literature. Specifically, we present:

- Over GPTSs:
  - Congruence format for probabilistic bisimilarity (Section 7.1).
  - Non-expansiveness format for (approximated) \( \varepsilon \)-bisimulations (Section 10.1).

- Over RPTSs:
Table 1

<table>
<thead>
<tr>
<th></th>
<th>GPTS</th>
<th>RPTS</th>
<th>NPTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similarity</td>
<td></td>
<td></td>
<td>Positive PGSOS [33]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(precongruence)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Def. 32 Theorem 3</td>
</tr>
<tr>
<td>Ready similarity</td>
<td>PRBB-safe [36,91]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(congruence)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Def. 36 Theorem 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bisimilarity</td>
<td>GTSS [87]</td>
<td>RTSS [87]</td>
<td>PGSOS [17,40,42]</td>
</tr>
<tr>
<td></td>
<td>(congruence)</td>
<td></td>
<td>(congruence)</td>
</tr>
<tr>
<td></td>
<td>Def. 30 Theorem 1</td>
<td></td>
<td>Def. 32 Theorem 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Branching bisimilarity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRBB-safe [36,91]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(congruence)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Def. 36 Theorem 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rooted branching bisimilarity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Non-extensive [34]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Def. 45 Theorem 9</td>
</tr>
<tr>
<td></td>
<td>(non-expansiveness)</td>
<td>(non-expansiveness)</td>
<td>(non-expansiveness)</td>
</tr>
<tr>
<td></td>
<td>Def. 38 Theorem 5</td>
<td>Def. 40 Theorem 6</td>
<td>Def. 44 Theorem 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Non-extensive [34]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Def. 46 Theorem 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Lipschitz [34]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Def. 47 Theorem 11</td>
</tr>
<tr>
<td>Bisimilarity metric</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Congruence format for probabilistic bisimilarity (Section 7.2).
- Non-expansiveness format for (approximated) ε-bisimulations (Section 10.2).

* Over NPTs:

- Congruence formats for strong bisimilarity, similarity, and ready similarity (Section 8.1), and the congruence formats for branching bisimilarity and its rooted version (Section 8.2).
- Non-expansiveness format for (approximated) ε-bisimulations (Section 11).
- Non-extensiveness, non-expansiveness, and Lipschitz continuity formats for bisimilarity metric (Section 13).

In Table 1 we present a recap of all the formats that we will present in the upcoming sections.

1.2. Organisation of contents

To the mere purpose of highlighting the various contents of this survey, we have divided it into five parts.

Part I, which consists of Sections 2–6, includes all the background notions that are necessary to the presentation of the formats that are covered in the survey. In detail, Section 2 reviews all the basic notions of the SOS framework given for classic process algebras. Those notions are then extended to the probabilistic setting in Sections 3–6. In Section 3 we present the three probabilistic extensions of LTs for which formats have been proposed: GPTSs, RPTSs, and NPTSs. The description of how these models are generated is then given in Section 4. Section 5 introduces all the equivalences, approximated equivalences, and metrics whose compositional properties, presented in Section 6, have been characterised by means of the formats discussed in the remainder of the survey.

Part II includes Sections 7–9, and covers the formats proposed to guarantee the congruence property of various (bi)simulation relations. Specifically, in Section 7 we present the congruence formats for bisimilarity over the GPTS and RPTS models. In Section 8, we consider the NPTS model and we review the congruence formats for (bi)simulation relations, and the congruence formats for (rooted) branching bisimilarity. Section 9 then recaps and compares the formats presented in this part.

Part III deals with approximated equivalences: Section 10 covers the non-expansiveness formats for ε-bisimulations over GPTSs and RPTSs, while Section 11 those for NPTSs. The part is concluded by Section 12 that highlights the principal features of the three formats.

Part IV consists of Sections 13 and 14, in which we present and compare all the continuity formats studied for the bisimilarity metric over NPTSs: the non-extensiveness, non-expansiveness, and Lipschitz continuity formats.
Part V concludes the paper with a brief discussion, in Section 15, on other probabilistic models existing in the literature, but for which no format has been proposed, and an informal presentation, in Section 16, of formats proposed outside of the process algebraic setting.

Part 1. Background

2. Process algebras and Structural Operational Semantics

In order to make our survey accessible to everyone, besides keeping it self-contained, we devote this section to a recap of standard notions in classic process algebras and in the structural operational semantics (SOS) framework.

Process algebras [20] are a classic tool for reasoning about the behaviour of concurrent, distributed systems. In this setting, systems are specified as processes, which are syntactically built as closed terms over an algebraic structure, called term algebra, that uses a set of operators to construct complex terms by combining simpler terms. The SOS framework [100] was introduced to provide a model for process algebras, by equipping processes with an operational semantics that describes their computation steps and the state-changes in a step-by-step fashion. This is achieved through the generation of a labelled transition system (LTS) [83] whose states are processes, and whose transitions between states are labelled with meta-variables, called actions, that abstract the computational steps inducing that state-transformation.

Our order of business for the remainder of this section is to formally introduce these concepts.

2.1. Term algebras

A signature \( \Sigma \) is a set of operators (or function symbols) each characterised by an arity (or rank), i.e., a number of arguments. Operators of arity 0 are called constants, while operators of arity one and two are called unary and binary, respectively. We let \( f, g, h, \ldots \) range over \( \Sigma \).

**Definition 1** (Process terms). Let \( \Sigma \) be a signature, and let \( \mathcal{V}_p \) be a countable set of process variables, ranged over by \( x, y, z, \ldots \), disjoint from \( \Sigma \). The set \( \mathcal{T}(\Sigma, \mathcal{V}_p) \) of the (open) process terms over \( \Sigma \) and \( \mathcal{V}_p \), ranged over by \( t, t', \ldots \), is the least set such that:

- \( x \in \mathcal{T}(\Sigma, \mathcal{V}_p) \), for all variables \( x \in \mathcal{V}_p \), and
- \( f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V}_p) \) if \( f \in \Sigma \) is an operator of rank \( n \) and \( t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V}_p) \).

A process term is closed if it does not contain any variable. We let \( \mathcal{T}(\Sigma) \) denote the subset of the closed process terms in \( \mathcal{T}(\Sigma, \mathcal{V}_p) \). Closed process terms are also called processes.

It is common in the literature to write \( f \) instead of \( f() \) when \( f \in \Sigma \) is a constant, and to use infix notation for binary operators, e.g., one writes \( t_1 + t_2 \) instead of \( +(t_1, t_2) \). Process terms will henceforth simply called terms. The reason why \( \mathcal{V}_p \) is called the set of process variables will become clear in Section 4.

**Example 1.** Assume a set of action names \( \text{Act} \). These actions are meta-variables that describe, in an abstract fashion, what happens in one computation step, including the possible interactions of the process with the environment. Let \( \overline{\text{Act}} \) denote the set of action co-names, i.e., \( \overline{\text{Act}} = \{ \overline{a} \mid a \in \text{Act} \} \). As usual, we postulate that \( \overline{a} = a \) and \( a \neq \overline{a} \) for all \( a \in \text{Act} \). Then, we let \( \mathcal{A} = \text{Act} \cup \overline{\text{Act}} \cup \{ \tau \} \), for a special action name \( \tau \notin \text{Act} \), known as the silent, or internal, action. Let \( a, b, \ldots \) range over \( \mathcal{A} \). We remark that \( \overline{\tau} \) is undefined.

The syntax of (recursion-free) Milner’s Calculus of Communicating Systems (CCS) [96] over \( \mathcal{A} \) is generated by the following grammar:

\[
\begin{align*}
t & ::= \text{nil} \mid x \mid a.t \mid t + t \mid t \parallel_{\text{ccs}} t \mid t[f] \mid t \\land A
\end{align*}
\]

where \( x \in \mathcal{V}_p \), \( a \in \mathcal{A} \), \( A \subseteq \text{Act} \), and \( f : \mathcal{A} \to \mathcal{A} \) is a relabelling function such that \( f(\tau) = \tau \) and \( f(\overline{a}) = \overline{f(a)} \) for all \( a \in \text{Act} \). Hence, the signature of CCS consists in: (i) the constant \text{nil}; (ii) the unary operators action prefixing (\( a \cdot \)), relabelling (\( -f \)), and restriction (\( \cdot A \)); (iii) and the binary operators nondeterministic alternative composition (\( + \cdot \)), and parallel composition (\( \cdot \parallel_{\text{ccs}} \)). These operators allow us to model the following process behaviours. \text{nil} is the idling process, i.e., the process that cannot perform any action. Process \( a.t \) performs action \( a \) and then proceeds its computation as process \( t \). Process \( t_1 + t_2 \) behaves either as \( t_1 \) or as \( t_2 \), discarding the process that is not chosen. The choice is nondeterministic in the sense that it is not possible to establish a priori which process will be selected (as the choice might depend on an interaction with the environment). The parallel components in \( t_1 \parallel_{\text{ccs}} t_2 \) can either interleave their behaviour, or they can synchronise (communicate) over actions \( a \) and \( \bar{a} \). The result of a synchronisation is the execution of a \( \tau \) action by \( t_1 \parallel_{\text{ccs}} t_2 \). We remark that, since \( \overline{\tau} \) is undefined, communication cannot occur over silent actions. Relabelling assumes a function \( f : \mathcal{A} \to \mathcal{A} \) (which behaves like the identity over \( \tau \) and is commutative with \( \overline{\cdot} \)), which is applied to the actions performed by the argument process. The restriction operator is used to inhibit actions by a process: \( \land A \) behaves as \( \tau \) but the actions in \( A \parallel \overline{\mathcal{A}} \) are forbidden.

**Remark 1.** When writing CCS terms, sometimes trailing occurrences of \text{nil} will be omitted, i.e., we may simply write, e.g., \( a + x \) instead of \( a \cdot \text{nil} + x \), for any \( a \in \mathcal{A} \), \( x \in \mathcal{V}_p \).
2.2. Labelled transition systems

The semantics of processes is defined in terms of (labelled) transitions modelling the computation steps. A transition of the form $\mathit{t} \xrightarrow{a} \mathit{t}'$ expresses that process $\mathit{t}$ performs a computation step that is described by the action-label $a$, and then proceeds its computation as specified by process $\mathit{t}'$. The collection of all the transitions describing the behaviour of processes is called a labelled transition system (LTS) [83].

**Definition 2 (LTS, [83]).** A labelled transition system (LTS) is a triple $(\mathit{T}(\Sigma), \mathcal{A}, \mathcal{T})$, where $\mathit{T}(\Sigma)$ is a set of processes (or states), $\mathcal{A}$ is a set of action labels, and $\mathcal{T} \subseteq \mathit{T}(\Sigma) \times \mathcal{A} \times \mathit{T}(\Sigma)$ is a (labelled) transition relation.

Following the notation conventions from the literature, we use $\mathit{t} \xrightarrow{a} \mathit{t}'$ in lieu of $(\mathit{t}, \mathit{a}, \mathit{t}') \in \mathcal{T}$. Process $\mathit{t}$ is called the source of the transition, whereas $\mathit{t}'$ is called the target. Moreover, for each $\mathit{t} \in \mathit{T}(\Sigma)$ and $\mathcal{A} \subseteq \mathit{A}$, we write $\mathit{t} \xrightarrow{a} \mathit{t}'$ if $\mathit{t} \xrightarrow{a} \mathit{t}'$ holds for some $\mathit{t}'$, and $\mathit{t} \not\xrightarrow{a}$ otherwise. To improve readability, and as common practice, in the examples we will present the behaviour of a process as the process graph, having processes as vertices, and transitions as edges, that can be constructed from the LTS.

**Example 2.** In Fig. 1 we depict the process graph obtained from the LTS $(\mathit{T}, \mathcal{A}, \mathcal{T})$ with: (i) $\mathit{T} = \{\mathit{nil}, \mathit{t}, \mathit{t}'\}$, with $\mathit{t} = \mathit{a} \mathit{t}'$ and $\mathit{t}' = \mathit{b} + \mathit{c}$; (ii) $\mathcal{A} = \{a, b, c\}$; (iii) $\mathcal{T} = \{(\mathit{t}, \mathit{a}, \mathit{t}'), (\mathit{t}', \mathit{b}, \mathit{nil}), (\mathit{t}', \mathit{c}, \mathit{nil})\}$. ◀

2.3. Transition system specifications

In order to show how an LTS is generated from the specification of processes, we need to introduce the notions of transition system specification and substitution.

Assume a signature $\Sigma$, a set of process variables $\mathbf{V}_p$, and a set of action labels $\mathcal{A}$.

A positive literal (respectively, negative literal) is an expression of the form $\mathit{t} \xrightarrow{a} \mathit{t}'$ (respectively, $\mathit{t} \not\xrightarrow{a}$), for any $\mathit{t}, \mathit{t}' \in \mathit{T}(\Sigma, \mathbf{V}_p)$ and $\mathcal{A} \subseteq \mathcal{A}$. Literals $\mathit{t} \xrightarrow{a} \mathit{t}'$ and $\mathit{t} \not\xrightarrow{a}$ are said to deny each other.

**Definition 3 (TSS, [75]).** A SOS rule has the form $\text{premise} \quad \text{conclusion}$, where premises is a set of literals and conclusion is a positive literal. A transition system specification (TSS) has the form $(\Sigma, \mathcal{A}, R)$, with $R$ a set of rules.

**Example 3.** The SOS rules for the CCS operators introduced in Example 1 are reported in Fig. 2. ◀

A substitution over $\mathbf{V}_p$ and $\Sigma$ is a mapping $\sigma : \mathbf{V}_p \rightarrow \mathit{T}(\Sigma, \mathbf{V}_p)$ from variables to terms, and it is called a closed substitution if all variables are mapped to closed terms, i.e. $\sigma(x) \in \mathit{T}(\Sigma, \mathbf{V}_p)$ for all $x \in \mathbf{V}_p$. Substitutions are extended from variables to terms, literals, and rules by element-wise application. In particular, a closed substitution instance of a literal (respectively, rule) is called a closed literal (respectively, closed rule).

The intuitive meaning of SOS rules is that if the conditions expressed by the literals in the premises are met by the processes substituted to the terms occurring in them, then the closed substitution instance of the conclusion can be derived. Formally, an LTS is derived from a TSS through the notion of proof.

**Definition 4 (Proof, [70]).** A proof from a TSS $T = (\Sigma, \mathcal{A}, R)$ of a closed literal $l$ is a well-founded, upwardly branching tree, with nodes labelled by closed literals, such that the root is labelled $l$ and, if $l'$ is the label of a node $q$ and $H$ is the set of labels of the nodes directly above $q$ (i.e., the labels of the children of node labelled $l'$), then:

- either $l'$ is positive and $\frac{H}{l}$ is a closed substitution instance of a rule in $R$,
- or $l'$ is negative and for each closed substitution instance of a rule in $R$ whose conclusion denies $l'$, a literal in $H$ denies one of its premises.

A closed literal $l$ is provable from $T$, notation $T \vdash l$, if there exists a proof from $T$ of $l$. The LTS induced by $T$ has as transitions the closed literals that are provable from $T$. 

![Diagram](image-url)
Remark 2. As noticed in [74], it might happen that if some rules in the TSS have negative premises, then it is not straightforward to associate an LTS to it, since, for instance, the TSS may allow us to derive transitions that deny each other. We refer the interested reader to [70] for a thorough discussion of this issue and the possible solutions to it. We limit ourselves to remark that all the TSSs that we consider in this survey are meaningful, and each one of them generates a unique model corresponding to the LTS whose transition relation contains exactly the closed positive literals that are provable from it.

The notion of disjoint extension for a TSS allows us to introduce new operators without affecting the behaviour of those already specified.

Definition 5 (Disjoint extension). A TSS $T' = (\Sigma', A, R')$ is a disjoint extension of a TSS $T = (\Sigma, A, R)$, written $T \sqsubseteq T'$, if $\Sigma \subseteq \Sigma'$, $R \subseteq R'$ and $R'$ introduces no new rule for any operator in $\Sigma$.

2.4. Behavioural equivalences

So far, we have seen that systems are specified as processes whose behaviour is modelled as an LTS, and how such an LTS is generated. A natural question is then to verify whether the behaviour of a process is equal to the intended behaviour of the system. Hence, we need to establish a criterion to compare the behaviour of processes. Behavioural relations have been proposed with this exact purpose: abstract from unwanted details of process behaviour and identify those processes that interact “similarly” with the environment. They consist of behavioural equivalences and behavioural preorders, where a preorder is a relation that is reflexive and transitive, while an equivalence is a preorder which is also symmetric. Equivalences are usually used to establish whether two processes are indistinguishable for behaviour, whereas preorders are mostly used to establish process refinements with respect to behaviour.

Due to the large number of properties which may be relevant in the analysis of systems, many different theories of equivalences were proposed. Essentially, two processes are considered equivalent if: (i) they either perform the same sequences of actions (trace approach); (ii) or they perform the same sequences of actions, and after each sequence they are ready to accept the same sets of actions (testing approach); (iii) or they perform the same sequences of actions, and after each sequence they exhibit, recursively, the same behaviour (bisimulation approach). Moreover, in an orthogonal way, the strong approach to equivalences and the weak one differ since only in the first case the processes are discriminated also on the basis of their ability to perform internal computation steps, such as communications between subsystems, that are not visible to the external observers. For an exhaustive presentation of behavioural equivalences, preorders, and their relations, we refer the interested reader to van Glabbeek’s linear time - branching time spectra [69,71].

2.5. Reasoning compositionally: congruences and formats

Equipping processes with a semantics is not the only application of the SOS framework. One of the main concerns in the development of a meta-theory of process languages is to guarantee their compositionality, namely to prove the compatibility of the language operators with the behavioural relation chosen for the application context. In algebraic terms, this compatibility is known as the congruence (resp. precongruence) property of the considered behavioural equivalence (resp. preorder). This property, formally introduced later in Definition 27, guarantees that the substitution of a subcomponent of the system with an equivalent one does not affect the behaviour of the system.

The SOS framework plays a crucial role in supporting the compositional reasoning and verification: a rule format is a set of syntactical constraints over SOS rules ensuring the desired semantic properties of the transition system derived from them. A rule format becomes a specification format if some constraints are imposed also on the set of the rules. Thus, one can prove useful results, as the (pre)congruence property, for a whole class of languages at the same time. These formats are also referred to as congruence formats, and we refer the interested reader to the survey papers [1,18] for a presentation of existing formats for classic behavioural relations. For instance, the De Simone format [106] ensures that trace equivalence is a congruence; the GSOS format [27] works for bisimilarity; and in [26] the ntysf/nxts format [74] is reduced to the ready trace format and its variants to guarantee that decorated trace preorders are precongruences.
Example 4. As an example of a rule format, we present the Grand SOS (or simply GSOS) format of Bloom, Istrail, and Meyer [27]. Our choice fell on it because all the formats that we will discuss in this survey can be considered as extensions of this format to the probabilistic setting.

We say that an SOS rule is in GSOS format if it has the form

\[
\left\{ x_i \xrightarrow{a_i} y_{ij} \mid 1 \leq j \leq m_i, 1 \leq i \leq n \right\} \quad \left\{ x_i \xrightarrow{b_{jk}} y_{kl} \mid 1 \leq k \leq n_j, 1 \leq i \leq n \right\}
\]

\[
f(x_1, \ldots, x_n) \xrightarrow{a} t
\]

where: (i) \( n \) is the arity of \( f \in \Sigma \); (ii) \( t \in T(\Sigma, V_p) \); (iii) \( m_i, n_i \in \mathbb{N} \) for each \( i = 1, \ldots, n \); (iv) the variables \( x_i \) and \( y_{ij} \) appearing in the premises are all distinct and the only variables that occur in the rule.

Notice that all the rules for CCS operators, presented in Fig. 2, are in GSOS format. ▪

3. Models of probabilistic processes

Probabilistic process algebras [7,45,80] extend classic process algebras, by adding suitable operators allowing us to express probability distributions over sets of possible events or behaviours.

Example 5. We recall some operators from [6,7,17,42,47,49] that we will use throughout the survey to exemplify various definitions and results. The non-probabilistic operators are inherited from CCS (given in Example 1), Communicating Sequential Processes (CSP) [79] and the Algebra of Communicating Processes (ACP) [18,19]. Here we introduce these operators only informally, and the SOS rules formalising their semantics will be given in next sections. We remark that our aim is not to consider a single signature including all these operators, but rather to showcase the most common (and best known) operators from the literature, and use them in the upcoming examples.

Besides CCS’s idling process nil, examples of constants are the ACP-like process \( \eta \), which performs action \( \sqrt{\cdot} \) and terminates with success, and the ACP-like process \( a \), which performs the action \( a \in A \) and will proceed as \( \eta \). This description assumes that \( \sqrt{\cdot} \) is a special element in \( A \) denoting the successful termination.

Probabilistic CCS-like/CSP-like action prefixing allows for composing an action with a probabilistic choice of processes. In detail, process \( a \odot_{[p]} b \) performs the action \( a \) and continues as process \( b \) with probability \( p \). We may write \( a.(t_1 \odot_{[p]} t_2) \) if \( n = 2, p_1 = p \) and \( p_2 = (1 - p) \). Then, we may write \( a.t_1 \) if \( n = 1 \) and \( p_1 = 1 \). In order to describe sequences of behaviours, in CCS and CSP we use action prefixing, whereas in ACP sequential composition is used: \( t_1 ; t_2 \) starts running as \( t_1 \) and will proceed as \( t_2 \) after the successful termination by \( t_1 \). Finite iteration \( r^* \) is used as a syntactic sugar for \( t; \ldots ; t \) with \( n \) occurrences of \( t \).

Infinite iteration \( r^\omega \) let an infinite number of copies of \( r \) run one after the other. However, if \( t \) can terminate with success, also \( r^* \) and \( r^\omega \) can. The unary Kleene star process \( r^* \) from [13] differs from \( r^\omega \) in that it can terminate with success regardless of whether \( t \) can do the same or not.

The ACP-like merge \( t_1 \parallel t_2 \) is a form of parallel composition where the two processes \( t_1 \) and \( t_2 \) can interleave their actions and can also synchronise. In order to achieve synchronisation, we need a partial function \( \times : A \times A \to A \) mapping some pairs of actions describing the steps of \( t_1 \) and \( t_2 \) to the action describing their synchronisation. If \( a_1 \times a_2 \) is undefined, then the steps described by \( a_1 \) and \( a_2 \) cannot synchronise. In the CSP-like parallel composition \( t_1 \parallel_{[p]} t_2 \) the processes \( t_1 \) and \( t_2 \) are forced to synchronise on actions in \( A \) and interleave actions not in \( A \), where \( A \) is a subset of \( A \). We remark that CSP-like parallel composition allows for multi-party communications, whereas in CCS communication is a two-party handshake (cf. Fig. 2). The probabilistic interleaving \( t_1 \parallel_{[p]} t_2 \) interleave the actions of the process arguments \( t_1 \) and \( t_2 \) with respect to a probability weight \( p \). In detail, whenever a given action \( a \) can be performed by both \( t_1 \) and \( t_2 \), then \( t_1 \) makes \( a \) with probability \( p \) and \( t_2 \) with probability \( 1 - p \). If \( a \) can be performed only by \( t_i \) (resp. \( t_2 \) then \( t_1 \parallel_{[p]} t_2 \) performs the \( a \)-move of \( t_i \) (resp. \( t_2 \). The probabilistic alternative composition \( t_1 +_{[p]} t_2 \) differs from \( t_1 \parallel_{[p]} t_2 \) in that the process that does not move is discarded.

The replication \( !_A t \) creates an arbitrary number of copies of argument \( t \) that will run in parallel. More precisely, they are combined by using operator \( \parallel_A \). In its probabilistic version \( !_{[p,A]} t \) from [97], replication is realised with probability \( 1 - p \). Clearly, replication and probabilistic replication may each exploit a different version of parallel composition, without any conceptual difference.

The priority operator \( \theta^p_t \) allows the argument process to perform \( b \) only if it cannot perform \( a \), whereas all other actions that can be performed by the argument process \( t \) can be performed also by \( \theta^p_t(t) \). ▪

As transitions are used to model the behaviour of processes, it is clear that, in the probabilistic setting, we need to modify the relations of the form \( t \xrightarrow{a} t' \) in order to capture also the probabilistic behaviour. However, several approaches exist, each describing the probabilistic behaviour from a different perspective. Hence, in the literature we can find a wealth of proposals of probabilistic extensions of LTSs. In this section, we recall the three models for probabilistic processes for which rule formats (that are the crux of our survey) have been studied, namely the generative (Section 3.1), the reactive (Section 3.2), and the nondeterministic and probabilistic (Section 3.3) models. For completeness, the interested reader can find an informal description of the probabilistic models for which no rule format has been given in the literature in Section 15.
3.1. The generative model of probabilistic processes

In the generative, or full, model of probabilistic processes \([12,15,29,72,108]\), for each process a single probability distribution is ascribed to all its moves. In \([72]\) the mechanistic view of the generative model is explained in terms of button pushing experiments. Intuitively, the environment, or observer, experiments on the process by attempting to depress the buttons that constitute its interface. Buttons correspond to action labels. Briefly, the observer tries to push all buttons at a time. The experiment succeeds if there are buttons that are unlocked and, in that case, the process will decide according to a prescribed probability distribution which unlocked button will go down. Then, the process is ready for the next experiment.

In the generative model, computation steps are described by transitions of the form \(t \xrightarrow{a,p} t'\), expressing that process \(t\) has probability \(p\) to perform an action described by \(a\) and to reach the process \(t'\). As argued in \([72]\), \(i\) is an index allowing for distinguishing \(t \xrightarrow{a,p}\) \(t'\) from another transition from \(t\) to \(t'\) labelled with \(a\) and having the same probability weight \(p\). We use \(\langle \rangle\) and \(\langle \rangle\) as brackets for multisets.

**Definition 6 (GPTS, \([12,15,29]\)).** A generative probabilistic transition system, or GPTS for short, is of the form \((\Sigma, A, I, \rightarrow)\), where: (i) \(\Sigma\) is a signature, (ii) \(A\) is a set of action labels, (iii) \(I\) is a set of indexes, and (iv) \(\rightarrow \subseteq \Sigma(\Sigma) \times A \times (0,1) \times I \times (\Sigma)\) is a transition relation such that, for all processes \(t \in \Sigma\):

\[
\sum \langle p \mid \exists a \in A, i \in I, t' \in \Sigma : (t, a, p, i, t') \in \rightarrow \rangle \in (0,1).
\]

Definition 6 requires that each process \(t \in \Sigma\) be semistochastic, namely the sum of the probability of its outgoing transitions, if there are any, sum up to \(1\). Other definitions exist in the literature, such as those in \([72,108]\) which admit that the sum of the probability of the transitions leaving from a process is a value \(0 \leq p < 1\), the interpretation being that in that process deadlock has probability \(1 - p\).

We will write \(t \xrightarrow{a,p} t'\) for \((t, a, p, i, t') \in \rightarrow\). As a further notation, for a set of action labels \(A \subseteq A\) and a value \(0 \leq p \leq 1\), we write \(t \xrightarrow{A,p} \) to denote that \(\sum \langle p \mid \exists a \in A, i \in I, t' \in \Sigma : (t, a, p, i, t') \in \rightarrow \rangle = p\), namely \(p\) is the overall probability mass of all transitions leaving from \(t\) and having label in \(A\). Notice that, in particular, \(t \xrightarrow{\emptyset,0} \) denotes that there is no transition with label in \(A\) leaving from \(t\). We write \(t \xrightarrow{\emptyset,0} \) when \(t \xrightarrow{A^\emptyset,0} \) for some \(p > 0\), and \(t \xrightarrow{A^\emptyset} \) if \(t \xrightarrow{A^\emptyset,0} \)

**Example 6.** In Fig. 3a we depict the GPTS of process \(t = (a.b +_{1/3} a.c) +_{1/2} (b +_{1/2} c)\).

3.2. The reactive model of probabilistic processes

In the reactive model of probabilistic processes \([72,88]\), the kind of action of a process is chosen nondeterministically, and, then, a probability distribution is ascribed to all moves of the process labelled with that action. If we reason in terms of button pushing experiment, in the reactive case the observer selects one of the buttons and tries to depress it. The experiments succeed if the selected button is unlocked. The choice of the button is completely under the control of the observer, therefore it is nondeterministic under the point of view of the process. More precisely, for the process this is an external nondeterminism. If the selected button goes down, the process makes an internal state transition according to a probability distribution that is associated with the process and that button and is ready for the next experiment.

**Definition 7 (RPTS, \([72,88]\)).** A reactive probabilistic transition system, or RPTS for short, is of the form \((\Sigma(\Sigma), A, I, \rightarrow)\), where: (i) \(\Sigma\) is a signature, (ii) \(A\) is a set of action labels, (iii) \(I\) is a set of indexes, and (iv) \(\rightarrow \subseteq \Sigma(\Sigma) \times A \times (0,1) \times I \times (\Sigma)\) is a transitions relation such that, for all processes \(t \in \Sigma\) and action labels \(a \in A\):

\[
\sum \langle p \mid \exists a \in A, i \in I, t' \in \Sigma : (t, a, p, i, t') \in \rightarrow \rangle \in (0,1).
\]

Definition 7 requires that, for each process, the probability of its outgoing transitions labelled with some action \(a \in A\), if there are any, sum up to \(1\). This is known as the property of reactivity.

Notations \(t \xrightarrow{a,p} \), \(t \xrightarrow{A,0} \) and \(t \xrightarrow{A,p} \) are defined for RPTSs as for GPTSs.

**Example 7.** In Fig. 3b we report the RPTS of process \(u = (a.c +_{1/5} a.b) + (b +_{1/4} b.d)\).

3.3. The nondeterministic and probabilistic model of processes

The nondeterministic and probabilistic model \([40,102]\) can be viewed as a generalisation of the reactive model. Besides the nondeterministic choice on the kind of action performed by the process, there is also a nondeterministic choice among several probability distributions over the moves by the process that are associated with that action. Reasoning in terms of button pushing experiments,
as in the reactive case the observer selects one of the buttons in the process interface. Again, for the process this is an external nondeterministic choice. If the button is unlocked and goes down, then the process selects one of the available reactions to that button pressing, namely there is an internal nondeterministic choice, and makes an internal state transition according to a probability distribution that is associated with the selected reaction.

The introduction of the nondeterministic and probabilistic model requires to deal explicitly with distributions over processes, namely mappings \( \pi : T(\Sigma) \to [0,1] \) with \( \sum_{t \in T(\Sigma)} \pi(t) = 1 \). We use \( D(T(\Sigma)) \) to denote the set of all probability distributions over \( T(\Sigma) \), and we let \( \pi, \sigma \) range over \( D(T(\Sigma)) \).

**Definition 8 (NPTS, [40]).** A nondeterministic probabilistic transition system, or NPTS for short, is a triple of the form \((T(\Sigma), A, \rightarrow)\), where:

(i) \( \Sigma \) is a signature,
(ii) \( A \) is a set of action labels, and
(iii) \( \rightarrow \subseteq T(\Sigma) \times A \times D(T(\Sigma)) \) is a transition relation.

We write \( t \rightarrow a \pi \) for \((t,a,\pi) \in \rightarrow\), and \( t \xrightarrow{a} \pi \) if \( t \rightarrow a \pi \) holds for no distribution \( \pi \).

Summarising, NPTSs allow us to model, at the same time, reactive behaviour, nondeterminism and probability: A process can react to several events coming from the environment, each represented by a label in \( A \); then, for each event, there may be a nondeterministic choice between several reactions, namely a set of equally labelled transitions; finally each transition takes to a probabilistic choice over processes.

The NPTS model is known also as the Segala’s model, since NPTSs were introduced in Segala’s thesis [102] (and some previous works [101,103]). However, in this survey we adopt the equivalent process algebraic formulation proposed in [40], since it allows for a simpler, albeit general, formalisation.

In the following, we will consider a particular set of distributions over processes:

- for a process \( t, \delta_t \) is the Dirac (or point) distribution defined by \( \delta_t(t) = 1 \) and \( \delta_t(t') = 0 \) for \( t' \neq t \);
- for an operator \( f \in \Sigma \) of rank \( n \) and distributions \( \pi_1, \ldots, \pi_n \) is the product distribution defined by \( f(\pi_1, \ldots, \pi_n)(f(t_1, \ldots, t_n)) = \prod_{i=1}^n \pi_i(t_i) \) and \( f(\pi_1, \ldots, \pi_n)(t) = 0 \) for all \( t \) not in the form \( f(t_1, \ldots, t_n) \);
- the convex combination \( \sum_{\pi_i} p_i \pi_i \) of a family of distributions \( \{ \pi_i \}_i \) with \( p_i \in [0,1] \) and \( \sum_{i=1}^n p_i = 1 \) is defined by \( (\sum_{\pi_i} p_i \pi_i)(t) = \sum_{i=1}^n p_i \pi_i(t) \) for all \( t \in T(\Sigma) \).

The support of a distribution \( \pi \), i.e., the set of processes \( t \) such that \( \pi(t) > 0 \), will be denoted by \( \text{supp}(\pi) \).

**Remark 3.** In this survey, we consider only distributions with a finite support.

**Example 8.** In Fig. 3c we report the NPTS of process \( v = a.(1/3)\cdot(1/3)v_1 \oplus (1/3)v_2 \oplus (1/3)v_3 ) + b.(\text{nil} \oplus 1/7 c_1) \), where \( v_1 = b, v_2 = a + c, v_3 = v_4 = c \). In particular, by letting \( x_1 = 1/3 \delta_b + 1/3 \delta_a + 1/3 \delta_c \) and \( x_2 = 1/7 \delta_{\text{nil}} + 6/7 \delta_c \), we have depicted the transition relation 

\[ \xrightarrow{\text{\cdots}} = \{((v,a,x_1), (v,b,x_2), (v_1,b,\delta_{\text{nil}}), (v_2,a,\delta_{\text{nil}}), (v_2,c,\delta_{\text{nil}}), (v_3,c,\delta_{\text{nil}}), (v_4,c,\delta_{\text{nil}})) \}. \]

The distributions, that are reached via the action-labelled transitions, are represented by means of dotted arrows having the probability weight assigned to the target process as label. For brevity, we use • to denote the distribution \( \delta_{\text{nil}} \).

4. Probabilistic transition system specifications

The theory of TSSs (see Section 2.3) has been extended to the probabilistic case. Clearly, the differences in the models presented in Section 3 are reflected on the corresponding specifications. We discuss the technical subtleties characterising them in Sections 4.1, 4.2, 4.3.

Before entering in the details of those specifications, we remark that the basic syntactic notions introduced in Section 2, like those of process term, substitution, and rule, remain unchanged in the probabilistic setting.

**Example 9.** Examples of terms are \( a.(x_1 \oplus 0.3 x_2) \), \( a.x_1 +_{0.5} x_2 \) and \( a.x_1 \parallel_{0.5} b.x_2 \). Then, by applying a substitution \( \sigma \) with \( \sigma(x_1) = a.\text{nil} \) and \( \sigma(x_2) = b.\text{nil} \) we get the processes \( a.(a.\text{nil} \oplus 0.3 b.\text{nil}) \), \( a.a.\text{nil} +_{0.3} b.\text{nil} \) and \( a.a.\text{nil} \parallel_{0.5} b.b.\text{nil} \).
4.1. Transition system specifications for the NPTS model

We start with the NPTS model, since the SOS rules for it are simpler than those for other models. What makes these rules special is that they use specific terms allowing us to express distributions over terms. Indeed, we have seen above that the peculiarity of the NPTS model is that the transitions take processes to distributions over processes. Hence, the literals constituting the premises and the conclusions of the SOS rules for this model should include a syntactic representation of distributions. To this end, distribution terms (or open distributions) have been introduced as expressions containing distribution variables that can instantiate to distributions. The set of distribution variables will be denoted by \( \mathcal{V}_d \), disjoint from \( \mathcal{V}_p \) and \( \Sigma \), and ranged over by \( \mu, \nu, \ldots \). We will use \( \zeta, \zeta' \) to range over \( \mathcal{V}_p \cup \mathcal{V}_d \).

**Definition 9 (Distribution term, \([39,40,42]\)).** For sets of process variables \( \mathcal{V}_p \) and distribution variables \( \mathcal{V}_d \), the set \( \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d) \) of the distribution terms over \( \sigma \), \( \mathcal{V}_p, \mathcal{V}_d \), ranged over by \( \theta, \theta', \ldots \), is the least set s.t.:

- \( \mu \in \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d) \), for all distribution variables \( \mu \in \mathcal{V}_d \);
- \( \delta_{\emptyset} \in \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d) \), for all terms \( t \in T(\Sigma, \mathcal{V}_p) \);
- \( \sum_{i \in I} p_{i} \theta_{i} \in \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d), \) if \( \theta_{i} \in \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d), \) \( p_{i} \in (0, 1] \) and \( \sum_{i \in I} p_{i} = 1 \);
- \( f(\theta_{1}, \ldots, \theta_{n}) \in \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d), \) if \( f \in \Sigma \) is an operator of rank \( n \) and \( \theta_{1}, \ldots, \theta_{n} \in \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d) \).

A substitution can be extended to a mapping \( \sigma : \mathcal{V}_p \cup \mathcal{V}_d \rightarrow \mathcal{T}(\Sigma, \mathcal{V}_p) \cup \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d) \) such that \( \sigma(x) \in T(\Sigma, \mathcal{V}_p) \), for all \( x \in \mathcal{V}_p \), and \( \sigma(\mu) \in \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d) \), for all \( \mu \in \mathcal{V}_d \). Then, \( \sigma \) is closed if \( \sigma(x) \in T(\Sigma) \), for all \( x \in \mathcal{V}_p \), and \( \sigma(\mu) \in \mathsf{DT}(\Sigma) \), for all \( \mu \in \mathcal{V}_d \). Notice that closed substitutions allow us to map distributions terms to distributions.

**Example 10.** Examples of distribution terms are \( 0.3 \delta_{\alpha,x} + 0.7 \delta_{y} \) and \( \delta_{\alpha,n} \| 0.5 \mu \). Then, by applying some substitution \( \sigma \) with \( \sigma(x) = \text{nil} \), \( \sigma(y) = b.nil \) and \( \sigma(\mu) = \delta_{\alpha,n} \), we get the distributions \( 0.3 \delta_{\alpha,n} + 0.7 \delta_{b,n} \) and \( \delta_{\alpha,n} \| 0.5 \delta_{b,n} \).

**Definition 10 (PSOS TSS, \([39,40,42]\)).** A Probabilistic SOS rule, or PSOS rule, has the form:

\[
\begin{array}{c}
\{ x_i \xrightarrow{a_i} \mu_{i,k} \mid i \in I, k \in K_i \} \quad \{ x_i \xrightarrow{b_{ij}} \mu_{j,l} \mid i \in I, j \in L_i \} \\
\end{array}
\]

where: (i) \( f \in \Sigma \) is an operator of rank \( n \); (ii) \( I = \{ 1, \ldots, n \} \) is a set of indices for the arguments of \( f \); (iii) \( K_i, L_j \) are finite index sets; (iv) \( a_{i,k}, b_{i,j}, \alpha, \beta \in \mathcal{A} \) are actions; (v) \( x_i \in \mathcal{V}_p \) are process variables; (vi) \( \mu_{i,k} \in \mathcal{V}_d \) are distribution variables; (vii) \( \theta \in \mathsf{DT}(\Sigma, \mathcal{V}_p, \mathcal{V}_d) \) is a distribution term. A PSOS transition system specification (PSOS TSS) is a triple \( P = (\Sigma, \mathcal{A}, R) \), where \( \Sigma \) is a signature, \( \mathcal{A} \) is a set of action labels and \( R \) is a set of PSOS rules.

Let \( r \) be a PSOS rule as in Definition 10. The literals \( x_i \xrightarrow{a_i} \mu_{i,k} \), \( x \xrightarrow{b_{ij}} \mu_{j,l} \) and \( f(x_1, \ldots, x_n) \xrightarrow{\theta} \theta \) are called, respectively, positive premises (\( \text{prem}(r) \)), negative premises (\( \text{npos}(r) \)), and conclusion (\( \text{conc}(r) \)). The set of all premises is denoted by \( \text{prem}(r) \). Then, the term \( f(x_1, \ldots, x_n) \) is the source (\( \text{src}(r) \)), the variables \( x_1, \ldots, x_n \) are the source variables (\( x_i \in \text{src}(r) \)), and the distribution term \( \theta \) is the target (\( \text{tgt}(r) \)).

**Example 11.** All rules in Table 2 are PSOS rules. These rules specify the operators whose intuitive meaning was introduced in Example 5. We need to add some technical comment. The first rule for \( r^{t+1} \) formalises that \( n - m \) copies of \( t \) terminate with success, one copy performs \( s \) and \( m \) copies remain active. The second and third rules for the considered operator are similar and consider the cases \( m = 0 \) and \( m = n \), respectively. Notice that the case \( m = n \) differs from the case \( m < n \) since the ability by \( t \) to terminate with success is not among the premises of the rule. The fourth rule for \( r^{t+1} \) captures the case where all \( n + 1 \) copies of \( t \) terminate. Function \( x \) used in ACP-like merge requires that \( a_1 \times a_2 = \sqrt{a} \) if and only if \( a_1 = \sqrt{a} = a_2 \) and that \( a_1 \times a_2 \) is undefined whenever \( a_1 = \sqrt{a} \neq a_2 \) or \( a_1 \neq \sqrt{a} = a_2 \). Operator \( |A| \) is defined assuming that \( \sqrt{a} \in \mathcal{A} \). Operator \( \langle f \rangle \) requires that \( f(a) = \sqrt{a} \) if and only if \( a = \sqrt{a} \). Operator \( \emptyset_k \) is defined assuming that \( a \neq \sqrt{a} \).

In order to derive a NPTS from a TSS, one can exploit the notion of supported model, which was given in the probabilistic setting in \([39,40,42]\) and derived from the classic notion from \([27,28,70]\). In \([39,40,42]\) the considered rules are more general than the PSOS rules in Definition 10. However, if we restrict to PSOS rules, the NPTS that is derived from a TSS contains precisely the closed literals that are provable according to Definition 4.

4.2. Transition rules for the generative model

The rules for the generative model are more complicated than those for NPTSs, since we have to specify how the probability weight and the index of the conclusion are derived from those of the positive premises.

Given a set \( I \subseteq \{ 1, \ldots, n \} \) such that \( I = \{ i_1, \ldots, i_{|I|} \} \) and \( i_1 < \cdots < i_{|I|} \), and the set of indexes \( \{ i_1, \ldots, i_{|I|} \} \subseteq I \), we let \( |I| \) denote the sequence of indexes \( i_1, \ldots, i_{|I|} \).
Table 2
Some operators of probabilistic process algebras specified in the NPTS style.

| \[ \eta \] & \[ \vec{\delta} \] & \[ \sum_{i=1}^{n} p_{\delta_i} \] |
|----------------|-------------------------|-------------------------|
| \[ a \wedge \delta \] & \[ a \Rightarrow \delta \] & \[ a \wedge \delta \] |
| \[ x_1 \perp x_2 \] & \[ x_1 \perp x_2 \] & \[ x_1 \perp x_2 \] |
| \[ \delta_1 \] & \[ \delta_2 \] & \[ \delta_3 \] |
| \[ x_1 \oplus y \] & \[ x_1 \oplus y \] & \[ x_1 \oplus y \] |
| \[ x_1 \rightarrow x_2 \] & \[ x_1 \rightarrow x_2 \] & \[ x_1 \rightarrow x_2 \] |
| \[ x_1 \rightarrow x_2 \] & \[ x_1 \rightarrow x_2 \] & \[ x_1 \rightarrow x_2 \] |
| \[ x_1 \rightarrow x_2 \] & \[ x_1 \rightarrow x_2 \] & \[ x_1 \rightarrow x_2 \] |
| \[ x_1 \rightarrow x_2 \] & \[ x_1 \rightarrow x_2 \] & \[ x_1 \rightarrow x_2 \] |

Definition 11 (Generative rule, [87]). A generative rule \( r \) for an operator \( f \in \Sigma \) has the form

\[ \frac{x_1 \cdots x_n : a_{i1} p_i}{x_1 \cdots x_n : a_{i1} p_i} \quad \frac{x_1 \cdots x_n : a_{i1} p_i}{x_1 \cdots x_n : a_{i1} p_i} \quad \frac{x_1 \cdots x_n : a_{i1} p_i}{x_1 \cdots x_n : a_{i1} p_i} \]

where: (i) \( I, J, H, K \) are sets of indices for the argument of \( f \) such that \( J \subseteq I \); (ii) \( a_i \in A \) are actions and \( A_j, B_j, C_k \subseteq A \) are sets of actions; (iii) \( a_i \notin A_j \) if \( i = j \); (iv) \( p_i \) is a variable with range \( (0, 1) \), \( i \) is a variable with range \( I \), and \( p_i^f \) is a variable with range \( [0, 1] \); (v) \( r \in T(\Sigma) \) is a term; (vi) \( w_r \in (0, 1) \) and is called the weight of \( r \).
The expressions \( x_{\mu,\nu} \) are called active premises and variables \( x_i \) are called active variables. The expressions \( x_{\mu,\nu} \) are called unneeded premises, where “unneeded” indicates that their probability \( p_j \) could be 0. The expressions \( x_{\mu,\nu} \) are called unquantified premises, where “unquantified” indicates that their probability is not relevant. The expressions \( x_k \) are called negative premises. The notions of conclusion, source, variable and target are as for the rules in Definition 10.

**Remark 4 (Double testing).** Definition 11 does not admit double testing, namely, two (or more) active premises cannot test moves of the same process argument. In [87], the authors explain how this feature can be added to generative rules, but given the heavy amount of technical subtleties necessary to do that, we decided not to report it here. Notice that double testing is admitted by Definition 10.

In Definition 11 we assume that the set of indexes \( I \) contains an index \( (r) \) for each rule \( r \) having a constant as source. Moreover, given any rule \( r \) with active variables \( \{x_i | i \in I\} \) and a set of indexes \( \{t_i | i \in I\} \subseteq I \), we assume that also \( (r,t_i)_{i \in I} \) is an index in \( I \).

**Example 12.** All rules in Table 3 and Table 4 are generative rules. They appear in two separate tables since the rules in Table 3 work also for the reactive case (presented in Definition 12 below). These rules specify operators from \([4,12,29,41,67,72]\). Even the simpler rules, like those for operators \( a \) and \( \eta \), look heavier than those in Table 2 due to the probability weight and the index in the transitions. We note that in the generative case, prefixing does not introduce any probabilistic behaviour. Operator \( +_p \) is used for this purpose. The rules for \( t_1 \) and \( t_2 \) in Table 3 are similar to those in Table 2. The rules for finite iteration \( r^{n+1} \) in Table 4 are partitioned into two sets: one contains the rules with a negative premise \( x_{\mu,\nu} \), and are applied if \( t \) cannot terminate with success; the other set contains the rules with an unquantified premise \( x_{\mu,\nu} \), and are applied when \( t \) can terminate with success. The first rule in Table 4 (actually, we have a rule for each action label \( a \in A \setminus \{\sqrt{1}\} \)) belongs to the first set and describes the case where \( t \) cannot terminate. If \( t \) can terminate, one applies the other two rules, which belong to the second set. If \( t \) terminates with probability \( p_1 \), then for any \( m \geq 0 \) it holds that \( p_1^{m+1} \) is the probability that \( m \) copies of \( t \) terminate with success one after the other. If this event is realised for \( m = 0 \), namely \( n \) copies of \( t \) terminate, then the remaining copy of \( t \) can either perform a non-\( \sqrt{1} \) action or terminate. This is the situation formalised in the second rule for \( r^{n+1} \). If that event is realised for \( m \geq 1 \), then if the first of the remaining copies of \( t \) performs a non-\( \sqrt{1} \) action then \( m \) copies of \( t \) remain active. This is the situation formalised in the third rule for \( r^{n+1} \). The rules for \( r^n \) and \( \sqrt{1} \) in Table 3 are similar to those in Table 2. In the generative case, the Kleene star, which is specified in Table 4, must be parametric with respect to a probability weight \( q \), the idea from [17] being that \( (r+p) \) behaves as \( t \) with probability \( 1 - q \) (first two rules) and terminates with success with probability \( q \) (third rule). The rules for probabilistic alternative composition \( t_1 +_p t_2 \) and probabilistic interleaving \( t_1 \parallel_p t_2 \) in Table 4 are partitioned in three sets, capturing the cases where both \( t_1 \) and \( t_2 \) can move, only \( t_1 \) can move, and only \( t_2 \) can move. Rules in the first set have positive premises for both arguments, rules in the second and third set use negative premises. Clearly, given a particular instance of process arguments \( t_1 \) and \( t_2 \), only the rules in one of those three sets will be applicable. Notice that in the generative case \( t_1 +_p t_2 \) and \( t_1 \parallel_p t_2 \) work differently with respect to the NPTS case formalised in Table 2. Essentially, in Table 2 the probabilistic choice applies only when both \( t_1 \) and \( t_2 \) intend to perform the same action, whereas in Table 4 the generative nature of probability imposes that when both \( t_1 \) and \( t_2 \) can move then \( t_1 \) moves with probability \( p \) and \( t_2 \) with probability \( 1 - p \) independently of the actions they intend to perform. Also operator \( \parallel \) (Table 4) is different with respect to its nondeterministic and probabilistic version. In detail, in the generative case we assume that \( a_1 \times a_2 \) is always defined whenever \( a_1 \neq \sqrt{1} \neq a_2 \). Without such an assumption, the probability of all pairs of actions performed by \( t_1 \) and \( t_2 \) that cannot be combined by \( x \) should be redistributed to the allowed synchronisations, thus making calculations really boring. However, the rules are anyway quite complicated, the reason being that a \( \sqrt{1} \) action by one process cannot synchronise with a non-\( \sqrt{1} \) action by the other, thus implying that the overall probability that the two processes attempt to synchronise a \( \sqrt{1} \) with a non-\( \sqrt{1} \) action should be redistributed to the probability of the allowed synchronisations. Such a normalisation of probability is done by exploiting unneeded premises, which allows us to quantify the overall probability of performing \( \sqrt{1} \) by the process arguments. The rules are partitioned into four sets, capturing the cases where both \( t_1 \) and \( t_2 \) can terminate with success, only \( t_1 \) can, only \( t_2 \) can, both cannot. The rules for relabelling and restriction are similar to those in Table 2, the only difference being that restriction \( t / A \) requires unneeded premises allowing for normalising the probability of transitions by taking into account that transitions from \( t \) with label in \( A \) are inhibited. Priority \( \mathcal{H}(i) \) requires both unquantified and unneeded premises: if the process argument \( t \) performs \( a \) then \( \mathcal{H}(i) \) cannot perform \( b \), therefore the overall probability weight of the \( b \)-moves by \( t \) must be used for normalisation.

<table>
<thead>
<tr>
<th>Table 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some operators of generative and reactive process algebras.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( a )</th>
<th>( \eta )</th>
<th>( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sqrt{1} )</td>
<td>( \mu,\nu )</td>
<td>( \eta )</td>
</tr>
<tr>
<td>( x_{\mu,\nu} )</td>
<td>( x_{\mu,\nu} )</td>
<td>( x_{\mu,\nu} )</td>
</tr>
<tr>
<td>( x_{\mu,\nu} )</td>
<td>( x_{\mu,\nu} )</td>
<td>( x_{\mu,\nu} )</td>
</tr>
<tr>
<td>( x_{\mu,\nu} )</td>
<td>( x_{\mu,\nu} )</td>
<td>( x_{\mu,\nu} )</td>
</tr>
</tbody>
</table>
We notice that the alternative composition $t_1 + t_2$ cannot be a generative operator. Moreover, operator $t_1 \parallel_A t_2$ could be defined, but whenever only one of the two processes performs an action in $A$ then its probability should be distributed to the allowed actions, thus requiring some complicate normalisation. Also CCS-like parallel composition $t_1 \parallel_{\text{ccs}} t_2$ may be defined, but that would require to deal with the following issue: since an $a$-transition can be performed autonomously or together with a $b$-transition of the partner...

---

**Table 4**

Some operators of generative process algebras.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1 + t_2$</td>
<td>Alternative composition</td>
</tr>
<tr>
<td>$t_1 \parallel_A t_2$</td>
<td>Distributed composition</td>
</tr>
<tr>
<td>$t_1 \parallel_{\text{ccs}} t_2$</td>
<td>CCS-like parallel composition</td>
</tr>
</tbody>
</table>

---

We notice that the alternative composition $t_1 + t_2$ cannot be a generative operator. Moreover, operator $t_1 \parallel_A t_2$ could be defined, but whenever only one of the two processes performs an action in $A$ then its probability should be distributed to the allowed actions, thus requiring some complicate normalisation. Also CCS-like parallel composition $t_1 \parallel_{\text{ccs}} t_2$ may be defined, but that would require to deal with the following issue: since an $a$-transition can be performed autonomously or together with a $b$-transition of the partner...
process, if synchronisation is possible then the probability of the $a$-transition should be partially assigned to the autonomous move and partially assigned to the handshaking.

We notice that unneeded premises are not required to derive the conclusion, meaning that $p'_j$ could be 0 for some $j \in J$. Unneeded premises are used only to compute the probability weight of the conclusion. More precisely, they allow for normalising probability, which is necessary in several operators of generative process algebras, such as restriction and priority. Indeed, the probability of the conclusion is the product of $r_{ij}$, which is the weight assigned to the rule $r_j$ by $\Pi_{j=1}^{\ell} r^j / \Pi_{j=1}^{\ell} - r^j$ which is the conditional probability that each process argument with index $i \in I$ performs $a_j$ under the assumption that each process argument with index $j \in J$ is not allowed to perform actions in $A_j$ (note that, coherently, clause (iii) in Definition 11 ensures that $a_j \not\in A_j$ when $i = j$). The choice of using sets of actions $A_j$ instead of single actions $a_j$ in unneeded premises is mandatory, since there are cases where an unneeded premise $a_j / p_j$ would not be sufficient for normalisation. For instance, if process $t_j$ is the argument of index $j$ and has two different moves $t_j \rightarrow^j t_j'$ and $t_j \rightarrow^j t_j''$, an unneeded premise $a_j / p_j$ would erroneously give rise to a normalisation factor $1 - 1/p_j$, whereas the unneeded premise $t_j \rightarrow^j a_j / p_j$ takes into account the probability of all $a_j$ moves of $t_j$ and gives rise to the correct normalisation factor $1 - 1/p_j$.

Unquantified premises do not contribute in computing the probability of the conclusion. They are necessary conditions for the application of rule $r$. We could replace unquantified premises by active premises. In fact, an unquantified premise $a_j / p_j$, with $B_h = \{h_1, \ldots, h_m\}$, may be replaced by active premises $a_j(b_h, y_{h,i}) / p_{h,i}$, with $1 \leq i \leq m$, where neither $p_{h,i}$ nor $y_{h,i}$ nor $i_{h,i}$ are mentioned in the conclusion. According to [87], distinguishing unquantified and active premises improves readability. Also negative premises $\neg a_j / p_j$, with $C_i = \{c_1, \ldots, c_m\}$, may be rewritten as sets of negative premises $\neg a_j / p_j \rightarrow^j \neg a_j / p_j$, mentioning only one action. The more compact notation was chosen for readability.

4.3. Transition rules for the reactive model

The notions of rule and TSS for the reactive case are similar to those for the generative case.

**Definition 12 (Reactive rule, [87]).** A reactive rule $r$ for an operation $f$ has the form

$$\begin{align*}
\{x_i \rightarrow^i t_i, y_i | i \in I\} & \{x_h \rightarrow^h | h \in H\} & \{x_k \rightarrow^k | k \in K\} \\
f(x_1, \ldots, x_n) & \rightarrow^r \Pi_{i \in I} ^{\ell} f (r(b_i)_{i \in I})
\end{align*}$$

where: (i) $I, H, K$ are sets of indices for arguments of $f$; (ii) $a_j, a \in A$ are actions and $B_h, C_k \subseteq A$ are sets of actions; (iii) $p_i$ is a variable with range $(0, 1]$ and $t_i$ is a variable with range $I$; (iv) $r \in T(\Sigma)$ is a term; (v) $w_r \in [0, 1]$ and is called the weight of $r$.

**Remark 5 (Double testing II).** As in the case of generative rules, Definition 12 does not admit double testing. Again, in [87] the authors explain how this feature can be added to reactive rules.

Since normalisation of probability is not required in the reactive case, Definition 12 does not need unneeded premises. Also in this case, the notions of supported transition and supported model for RTSSs are analogous to those presented for PSOS-TSSs.

**Example 13.** All rules in Table 3 and Table 5 are reactive rules. The rules for sequencing $t_1: t_2$, iterations $r^m, r^{m+1}$ and $r^+$, replication $\mid_l$, restriction $t \mid_A$ and priority $\Pi_{\ell}^j (t)$ are similar to those in Table 2, modulo the different shape of transitions. The rules for $r^{m+1}$ differ from those in Table 2 and Table 4. For any action label $a$, the rules for $r^{m+1}$ are partitioned in two sets, which are discriminated by the ability, or not, of performing $\sqrt{v}$ by $t$. In particular, if $t$ can terminate, an arbitrary number $m \in [0, n]$ of copies of $t$ terminate before the subsequent occurrence of $t$ makes the $a$ move. In order to guarantee the reactivity property, we assume that the probability that $m$ copies of $t$ terminate is $\Pi_{i=1}^{m} t_i$, for each $m \in [0, n]$. In this way, the probability weight $p$ of a transition $t \rightarrow^a t'$ is partitioned into $n + 1$ probability weights $\Pi_{i=1}^{n} t_i$ assigned to transitions taking $r^{m+1}$ to processes $t', t^1, \ldots, t^n$. For each action label $a$, the rules for probabilistic alternative composition $t_1 \rightarrow^a t_2$ are partitioned in three sets: rules applied when both $t_1$ and $t_2$ can perform $a$, rules applied if only $t_1$ can perform $a$ and rules applied if only $t_2$ can perform $a$. Probabilistic interleaving is similar. Relabelling requires care in order to avoid that mapping two or more action labels to the same action label breaks reactivity property. If the process argument $a$ can perform precisely $n$ actions that are mapped by $f$ to the same $b$, then the probability weight $p$ of a transition $t \rightarrow^a b'$ with $f(a) = b$ becomes $t_i/\pi$ in the transition $t[f] \rightarrow^b t'[f]$. Notice that $t_1 \parallel t_2$ cannot be defined in any trivial way, since synchronisation over different actions would give rise to $\tau$-transitions, thus implying that guaranteeing that the overall probability of $\tau$-transitions is 1 is really complicated. Also $t_1 \parallel t_2$ is not immediate to be defined, since actions not in $A$ may be performed by $t_1$ with overall probability 1, and the same for $t_2$, thus implying that the probability of the $a$-transitions by $t_1$ and $t_2$ should be weighted in some way.
5. Behavioural equivalences and metrics

Now that we have seen how the behaviour of probabilistic processes is modelled, we can discuss the tools that have been proposed for its analysis. In this section we introduce some notions of probabilistic behavioural equivalence, approximate equivalence and metric over probabilistic processes.

As previously outlined, when working with classic process algebras, we can use behavioural equivalences to identify those processes that interact “similarly” with the environment. Probabilistic behavioural equivalences [21,25,31,48,88,102,103,117] are their extensions to probabilistic processes: they formalise when two processes are able to engage the same interactions with the environment, with the same probability. Essentially, two processes are equivalent if they are indistinguishable by any external observer from both points of view on behaviour: qualitative and quantitative. However, it may happen that processes exhibit the same qualitative behaviours, but there are differences in the probability of their realisations, i.e., processes may have different probability to interact similarly with the environment. When using behavioural equivalences, even the tiniest difference in the probabilities causes the two processes to be considered totally unrelated. Approximate equivalences [3,4,52,53,67,112] and behavioural metrics [30,35,43,46,50,57,67,107] are less demanding than equivalences, since they relate two processes if they are approximately indistinguishable by any external observer. Technically, equivalences answer the question of whether two processes behave exactly in the same way or not, whereas approximate equivalences and behavioural metrics quantify the differences in their behaviours. Indeed, the approaches of approximate equivalences and metrics differ from each other. As we will see, in the former approach the distance between two processes is determined by the maximal difference that can be evaluated on a single computation step. Conversely, in the latter approach the behavioural distances arising along a computation path accumulate and are weighted by the probability of the realisation of that path.

Remark 6. The different approaches to behavioural relations discussed in Section 2.4, have been applied also to the probabilistic case, and spectra for equivalences [21,25] and metrics [35] have been proposed. In the following, we will restrict our attention to the bisimulation approach, since it is the only one for which rule formats have been developed.
5.1. Strong (bi)simulation

We introduce the notion of probabilistic (strong) bisimulation [72,88,102]. The intuition is that this equivalence relates two processes if they are able to mimic each other’s transitions and evolve to distributions that are, in turn, equivalent. This intuition is usually called the probabilistic bisimulation game.

The bisimulation game is formalised in different ways, depending on the probabilistic model that we consider. We start with the generative and the reactive models. To this end, we need to introduce the aggregate probability distribution function: $\gamma(t, a, T)$ computes the total probability to reach a set of processes $T$, through transitions labelled with action $a$, from a process $t$. Adopting the convention that summation over the empty set is 0, this function can be defined as follows:

**Definition 13 (Aggregate probability distribution function).** Assume a GPTS, or RPTS, $(T(\Sigma), A, I, \rightarrow)$. Then, the aggregate probability distribution function $\gamma : T(\Sigma) \times A \times 2^{T(\Sigma)} \rightarrow [0, 1]$ is the function given by:

$$\gamma(t, a, T) = \sum \{| p | \frac{a}{\pi} t \rightarrow t' \text{ and } t' \in T \} \quad \forall t \in T(\Sigma), \forall a \in A, \forall T \subseteq T(\Sigma).$$

Notice that the definition of $\gamma$ works for both the generative and the reactive models of processes. We will write $\gamma_G$ if we want to stress that we are dealing with a GPTS, and $\gamma_R$ for a RPTS.

**Remark 7.** In the case of NPTSs, we have implicitly an aggregate distribution function for each transition.

The notions of generative bisimulation and reactive bisimulation stem from [72,88]. For any equivalence relation $R$ over the set of processes $T(\Sigma)$, let $T(\Sigma)/R$ denote the set of equivalence classes induced by $R$.

**Definition 14 (Generative bisimulation [72] / reactive bisimulation [72,88]).** Let $(T(\Sigma), A, I, \rightarrow)$ be a GPTS. An equivalence relation $R \subseteq T(\Sigma) \times T(\Sigma)$ is a generative bisimulation if $(t_1, t_2) \in R$ implies:

$$\gamma_G(t_1, a, T) = \gamma_G(t_2, a, T) \quad \forall T \subseteq T(\Sigma)/R, \forall a \in A.$$  

Analogously, assuming a RPTS $(T(\Sigma), A, I, \rightarrow)$, an equivalence relation $R \subseteq T(\Sigma) \times T(\Sigma)$ is a reactive bisimulation if $(t_1, t_2) \in R$ implies:

$$\gamma_R(t_1, a, T) = \gamma_R(t_2, a, T) \quad \forall T \subseteq T(\Sigma)/R, \forall a \in A.$$  

**Generative bisimilarity** is the union of all generative bisimulations, it is, in turn, a generative bisimulation. We denote it by $\sim_G$, and write $t_1 \sim_G t_2$ for $(t_1, t_2) \in \sim_G$. **Reactive bisimilarity** is the union of all reactive bisimulations, it is, in turn, a reactive bisimulation. We denote it by $\sim_R$, and write $t_1 \sim_R t_2$ for $(t_1, t_2) \in \sim_R$.

Notice that the only difference between the two notions of bisimulation presented above is in the probabilistic model over which they are defined: a GPTS for generative bisimulations, and a RPTS for the reactive ones.

If we consider the model of NPTSs, the notion of bisimulation stems from [102]. In order to present it formally, we need to introduce the lifting of a relation over processes to a relation over distributions over processes. To this end, we rely on the notion of matching, also known as measure coupling [116].

**Definition 15 (Matching, [116]).** Assume a set $X$. A matching for a pair of distributions $(\pi, \pi') \in D(X) \times D(X)$ is a distribution over the product state space $\omega \in D(X \times X)$ with left marginal $\pi$, i.e. $\sum_{x \in X} \omega(x, x') = \pi(x)$ for all $x \in X$, and right marginal $\pi'$, i.e. $\sum_{x \in X} \omega(x, x') = \pi'(x')$ for all $x' \in X$. We let $\Omega(\pi, \pi')$ denote the set of all matchings for $(\pi, \pi')$.

**Definition 16 (Relation lifting).** The lifting of a relation over processes $R \subseteq T(\Sigma) \times T(\Sigma)$ is the relation over distributions $R' \subseteq D(T(\Sigma)) \times D(T(\Sigma))$ with $\pi R' \pi'$ whenever there is a matching $\omega \in \Omega(\pi, \pi')$ such that $t R t'$ for all processes $t, t'$ with $\omega(t, t') > 0$.

The notion of bisimulation for the NPTS model can be given together with those of two preorders, known as simulation and ready simulation. We opted for this choice since rule formats have been developed also for these preorders, and will be presented in Section 8.

**Definition 17 ((B)simulations, [102]).** Assume a NPTS $(T(\Sigma), A, \rightarrow)$. Then:

1. A relation $R \subseteq T(\Sigma) \times T(\Sigma)$ over processes is a simulation if whenever $t R t'$, for each transition $t \xrightarrow{a} \pi$ there is a transition $t' \xrightarrow{a} \pi'$ such that $\pi R' \pi'$.
2. A simulation $R$ is a ready simulation if whenever $t R t'$, $t \xrightarrow{a}$ implies $t' \xrightarrow{a}$.
3. A bisimulation is a symmetric simulation.
The union of all simulations (resp.: ready simulations, bisimulations) is the greatest simulation (resp.: ready simulation, bisimulation), called similarity (resp.: ready similarity, bisimilarity), and is denoted with \(\sqsubseteq\) (resp.: \(\sqsubseteq_r, \sim\)) and is a preorder (resp.: preorder, equivalence).

**Remark 8.** In the remainder of the survey, we use the terms bisimulation/bisimilarity when we refer to the NPTS model, generative bisimulation/bisimilarity when we refer to the GPTS model, and reactive bisimulation/bisimilarity when we refer to the RPTS model.

### 5.2. Branching bisimulation

The special silent action label \(\tau\) was introduced by Milner [96] to describe those computation steps that are not observable by the external environment, like internal communications between subsystems. The weak approach to equivalences abstracts from \(\tau\)-labelled transitions, meaning that processes are not discriminated by their ability to perform them or not. We will introduce weak equivalences only for NPTSs, since the only rule formats provided in literature for weak equivalences over probabilistic processes consider that model.

The weak approach is technically based on the notion of weak transition. Following [48], in order to introduce weak transitions in NPTSs one needs to lift the transition relation to a relation on distributions, called hyper-transition in [91,92]. We assume that \(\tau \notin \mathcal{A}\) and let \(\mathcal{A}_r = \mathcal{A} \cup \{\tau\}\). Moreover, we pursue in using \(a,b,...\) to range over \(\mathcal{A}\) and we will use \(a,\beta,...\) to range over \(\mathcal{A}_r\).

**Definition 18 (Lifted transition, [48]).** Assume an NPTS \((\Sigma, \mathcal{A}, \rightarrow)\). We define the relation \(\overset{\tau}{\rightarrow} : (\Sigma(\mathcal{A}) \cup D(\Sigma(\mathcal{A}))) \times \mathcal{A} \times (D(\Sigma(\mathcal{A})))\) from \(\rightarrow\) as follows:

- \(t \overset{\tau}{\rightarrow} \pi\) if either \(t \overset{\delta}{\rightarrow} \pi\) or \(\pi = \delta_i\);
- for \(a \in \mathcal{A}, t \overset{\delta}{\rightarrow} \pi\) if \(t \overset{a}{\rightarrow} \pi\);
- for \(\alpha \in \mathcal{A}_r, \pi \overset{\alpha}{\rightarrow} \pi'\) if \(\pi = \sum_{i \in \text{supp}(\pi)} \pi(t)\delta_i\).

We write \(t \overset{\tau}{\rightarrow} \pi\) for the reflexive-transitive closure of the relation \(\overset{\tau}{\rightarrow}\).

Notice that \(\pi \overset{\delta}{\rightarrow} \pi'\) with \(a \neq \tau\) requires \(t \overset{a}{\rightarrow} \pi\) for all \(t \in \text{supp}(\pi)\), whereas \(\pi \overset{\tau}{\rightarrow} \pi'\) can be derived independently from the ability of processes in \(\text{supp}(\pi)\) of performing \(\tau\).

branching bisimulation [73] is well known to be the weak equivalence that better preserves the branching properties of processes. We recall here the notion of probabilistic branching bisimulation as presented in [36,37]. The proposed formulation is equivalent to the original, scheduler-free, version introduced in [8,91].

**Definition 19 (Branching bisimulation, [36,37]).** Given an NPTS \((\Sigma(\mathcal{A}), \mathcal{A}, \rightarrow)\), a symmetric relation \(\mathcal{B} \subseteq \Sigma(\mathcal{A}) \times \Sigma(\mathcal{A})\) is a branching bisimulation if whenever \(t \mathcal{B} t'\) and \(t \overset{a}{\rightarrow} \pi\), then:

- either \(a = \tau\) and \(\pi \mathcal{B}^\dagger \delta_i\), or
- \(t' \overset{a}{\rightarrow} \pi'' \mathcal{B} \pi'\) with \(\delta_i \mathcal{B}^\dagger \pi''\) and \(\pi \mathcal{B}^\dagger \pi'\).

As it is well known from classic process algebras, to ensure compositionality with respect to nondeterministic choice operators, we need to impose the root condition, which requires that an initial \(\tau\)-transition is mimicked by a \(\tau\)-transition, as if they were observable actions.

**Definition 20 (Rooted branching bisimulation).** Assume an NPTS \((\Sigma(\mathcal{A}), \mathcal{A}, \rightarrow)\) and a branching bisimulation \(\mathcal{B}\) on \(\Sigma(\mathcal{A})\). A symmetric relation \(\mathcal{R} \subseteq \Sigma(\mathcal{A}) \times \Sigma(\mathcal{A})\) is a rooted branching bisimulation if:

whenever \(t \mathcal{R} t'\), if \(t \overset{a}{\rightarrow} \pi\) for \(a \in \mathcal{A}_r\), then there is a transition \(t' \overset{a}{\rightarrow} \pi'\) with \(\pi \mathcal{B}^\dagger \pi'\).

The union of all (rooted) branching bisimulations is the greatest (rooted) branching bisimulation, is denoted \(\approx_b\) (resp. \(\approx_{rb}\)) and called (rooted) branching bisimilarity. In NPTSs with no divergence, namely without any infinite sequence of \(\tau\)-transitions, \(\approx_b\) and \(\approx_{rb}\) are equivalence relations [37].

### 5.3. Approximate bisimulation

Approximate bisimulations [2,4,5,52–55,112] are parametric with respect to a threshold \(\epsilon \in [0,1]\), representing the maximal “acceptable” difference between processes, and are, thus, also called \(\epsilon\)-bisimulations. They relax the notion of bisimulation in that they relate two processes if their probability to reach a set of processes, that are in turn \(\epsilon\)-bisimilar, differs by at most \(\epsilon\). Approximate bisimulations have been characterised in operational terms [67], by a modal logic [52,112], and in terms of games [52].
Before giving formal definitions, we observe that two processes with quite different behaviours could be linked by a sequence of processes, which, pairwise, have only a small behavioural difference. For instance, it may happen that both, the behavioural distance between \( t_1 \) and \( t_2 \), and that between \( t_2 \) and \( t_1 \), are \( \epsilon \), whereas the distance between \( t_1 \) and \( t_1 \) is \( 2\epsilon \). Therefore, if one defines \( \epsilon \)-bisimulations as equivalence relations, then there may be \( \epsilon \)-bisimulations containing \((t_1, t_2)\) and other \( \epsilon \)-bisimulations containing \((t_2, t_1)\), but none can contain both these pairs, thus implying that \( \epsilon \)-bisimulations cannot be closed with respect to union, and no notion of “the greatest” \( \epsilon \)-bisimulation can be given. Conversely, if we drop the transitivity requirement from the definition of \( \epsilon \)-bisimulations, then there may be \( \epsilon \)-bisimulations containing both \((t_1, t_2)\) and \((t_2, t_1)\). Summarising, either we renounce to the property of closure with respect to union for \( \epsilon \)-bisimulations, or we renounce to define them as equivalence relations. The former is the approach in [3,4,53], where GPTs and RPTs are considered, the latter is the approach in [52,55,112], that deal with NPTs.

We start by introducing approximate bisimulations for the generative and reactive models.

**Definition 21** (Generative/reactive \( \epsilon \)-bisimulation, [3,4,53]). Let \( \epsilon \in [0,1] \). Let \((T(\Sigma), A, I, \rightarrow)\) be a GPTS. An equivalence relation \( R_\epsilon \subseteq T(\Sigma) \times T(\Sigma) \) is a generative \( \epsilon \)-bisimulation if \((t_1, t_2) \in R_\epsilon \) implies:

\[
\forall a \in A : \sum_{T \in T(\Sigma)/R_\epsilon} |\gamma_G(t_1, a, T) - \gamma_G(t_2, a, T)| \leq \epsilon.
\]

Analogously, assuming a RPTS \((T(\Sigma), A, I, \rightarrow)\), an equivalence relation \( R_\epsilon \subseteq T(\Sigma) \times T(\Sigma) \) is a reactive \( \epsilon \)-bisimulation if \((t_1, t_2) \in R_\epsilon \) implies:

\[
\forall a \in A : \sum_{T \in T(\Sigma)/R_\epsilon} |\gamma_R(t_1, a, T) - \gamma_R(t_2, a, T)| \leq \epsilon.
\]

The idea behind Definition 21 is that if two processes are equated by some generative, or reactive, \( \epsilon \)-bisimulation, then their behavioural distance is \( \leq \epsilon \). We remark that, as in the case of Definition 14, while the formulations of the two notions of approximate bisimulations are the same, the aggregate distribution functions are evaluated over different probabilistic models: a GPTS in the case of generative \( \epsilon \)-bisimulations, and a RPTS for the reactive ones. Since Definition 21 and Definition 14 coincide when \( \epsilon = 0 \), the distance between processes equated by some generative or reactive bisimulation is 0. Technically, as anticipated above, \( \epsilon \)-bisimulations defined as equivalences cannot be closed with respect to union, which implies that we cannot give any notion of “the greatest” generative, or reactive, \( \epsilon \)-bisimulation.

The approach in [52] for the model of NPTSs is different, since \( \epsilon \)-bisimulations are not defined as equivalence relations (as, in particular, they are not transitive). This different approach allows \( \epsilon \)-bisimulations to be closed with respect to union.

**Definition 22** (\( \epsilon \)-bisimulation, [52,55,112]). Assume any \( 0 \leq \epsilon \leq 1 \) and a NPTS \((T(\Sigma), A, \rightarrow)\). A reflexive and symmetric relation \( R \subseteq T(\Sigma) \times T(\Sigma) \) is an \( \epsilon \)-bisimulation if whenever \( t_1 R t_2 \) and \( t_1 \xrightarrow{a} \pi_1 \), then there exists a transition \( t_2 \xrightarrow{a} \pi_2 \) such that \( \pi_1 R \pi_2 \), where

\[
\pi_1 R \pi_2 \text{ iff } \text{ for all } T \subseteq T(\Sigma) \text{ we have } \pi_1(T) \leq \pi_2(R(T)) + \epsilon
\]

where \( R(T) = \bigcup_{t \in T} \{ t' \mid t \rightarrow t' \} \).

We call \( t_1 \) and \( t_2 \) (resp. \( \pi_1 \) and \( \pi_2 \)) \( \epsilon \)-bisimilar if \( t_1 R t_2 \) (resp. \( \pi_1 R \pi_2 \)) for some \( \epsilon \)-bisimulation \( R \). Notice that \( \epsilon \)-bisimulations are reflexive and symmetric but not necessarily transitive. \( \epsilon \)-bisimulations are closed under union. We denote the largest \( \epsilon \)-bisimulation, called \( \epsilon \)-bisimilarity, by \( \sim_\epsilon \). According to [52], \( \epsilon \)-bisimulations induce a pseudo-metric (pseudo-metrics are formally introduced in Definition 23 below) over the set of closed terms \( d : T(\Sigma) \times T(\Sigma) \rightarrow [0,1] \) with \( d(t, t') = \inf \{ \epsilon \in [0,1] \mid t \sim_\epsilon t' \} \), where \( \inf \emptyset = 1 \). We say that \( t \) and \( t' \) are within the approximation bisimulation distance \( \epsilon \) if \( d(t, t') = \epsilon \).

5.4. Bisimulation metric

The notion of bisimulation metric [30,46,50] bases on the idea that two processes can be at a given distance \( \epsilon < 1 \) only if they can mimic each other’s transitions and evolve to distributions that are, in turn, at a distance \( < \epsilon \). These distances are defined as pseudometrics. We start by recalling the notion of pseudometric and its lifting to pseudometrics over distributions.

**Definition 23** (1-bounded pseudometric). A 1-bounded pseudometric on a set \( X \) is a function \( d : X \times X \rightarrow [0,1] \) such that: (i) \( d(x,x) = 0 \), (ii) \( d(x,y) = d(y,x) \), and (iii) \( d(x,y) \leq d(x,z) + d(z,y) \), for all \( x, y, z \in X \). The kernel of a pseudometric \( d \) on \( X \) consists in the set of the pairs of elements in \( X \) that are at distance 0, namely \( \ker(d) = \{(x,y) \in X \times X \mid d(x,y) = 0 \} \).

In order to lift the definition of a pseudometric over processes to a pseudometric over distributions over processes, we rely on the notions of matching (Definition 15 above) and Kantorovich lifting.

**Definition 24** (Kantorovich metric, [81]). The Kantorovich lifting of a 1-bounded pseudometric \( d : T(\Sigma) \times T(\Sigma) \rightarrow [0,1] \) is the 1-bounded pseudometric \( \mathbf{K}(d) : D(T(\Sigma)) \times D(T(\Sigma)) \rightarrow [0,1] \) defined by
\( K(d)(\pi, \pi') = \min_{\omega \in \Omega(d, \pi, \pi')} \sum_{t, t' \in T(\Sigma)} \omega(t, t') \cdot d(t, t') \quad \forall \pi, \pi' \in D(T(\Sigma)). \)

**Remark 9.** Since we are considering only distributions having a finite support (Remark 3), the minimum in Definition 24 is well defined.

The Kantorovich distance \( K(d)(\pi, \pi') \) can be understood as the minimum expected value of the ground distance \( d \), over the supports of \( \pi \) and \( \pi' \), with respect to the distribution \( \omega \).

Bisimulation metrics are normally parametric with respect to a discount factor allowing us to mitigate the distance of future transitions \([44,50]\). Informally, any difference that can be observed only after a long sequence of computation steps has less impact than the differences that can be witnessed at the beginning of the computation. In our context, the discount factor is a value \( \lambda \in (0, 1] \) and the distance arising at step \( n \) is mitigated by \( \lambda^n \) (\( \lambda = 1 \) means no discount).

**Definition 25 (Bisimulation metric, [46]).** Let \( \lambda \in (0, 1] \). A pseudometric \( d : T(\Sigma) \times T(\Sigma) \to [0, 1] \) is a bisimulation metric if for all \( t, t' \in T(\Sigma) \) with \( d(t, t') < 1 \), we have

\[ \forall t \xrightarrow{a} \pi \exists t' \xrightarrow{a} \pi' \text{ such that } \lambda \cdot K(d)(\pi, \pi') \leq d(t, t'). \]

Note that bisimulation metrics are pseudometrics by definition. In [46] it is proved that for each discount \( \lambda \) the smallest bisimulation metric exists, which is called \( \lambda \)-bisimilarity metric and denoted by \( b^\lambda \). The kernel of \( b^\lambda \) induces an equivalence relation that coincides with bisimilarity, i.e., processes are at distance 0 if and only if they are bisimilar.

**Proposition 1 ([30,46]).** For any \( t, t' \in T(\Sigma) \), we have that \( b^\lambda(t, t') = 0 \) if and only if \( t \sim t' \).

**Remark 10.** As elsewhere in the literature, we shall use the term bisimulation metric in place of bisimulation pseudometric. We recall that in a metric the requirement \( d(x, x) = 0 \) in Definition 23 is strengthened by requiring that \( d(x, y) = 0 \) if and only if \( x = y \). This explains why one needs pseudometrics and not metrics over processes: with a pseudometric two processes that behave in the same way can be at distance 0; with a metric this is possible only if the two processes are syntactically the same.

Bisimulation metrics can be also characterised as prefixed points of a monotone bisimulation functional [46]. Let \( ([0,1]^{T(\Sigma) \times T(\Sigma)}, \leq) \) be the complete lattice of functions \( d : T(\Sigma) \times T(\Sigma) \to [0, 1] \), ordered by \( d_1 \leq d_2 \) if and only if \( d_1(t, t') \leq d_2(t, t') \), for all \( t, t' \in T(\Sigma) \).

**Definition 26 (Bisimulation metric functional).** Let \( B : [0,1]^{T(\Sigma) \times T(\Sigma)} \to [0,1]^{T(\Sigma) \times T(\Sigma)} \) be the functional defined, for all \( d : T(\Sigma) \times T(\Sigma) \to [0,1] \) and \( t, t' \in T(\Sigma) \), by:

\[ B(d)(t, t') = \sup_{\omega \in \Omega} \left\{ \sup_{\pi \xrightarrow{a} \pi_1} \lambda \cdot K(d)(\pi, \pi_1), \sup_{\pi' \xrightarrow{a} \pi_2} \inf_{\pi' \xrightarrow{a} \pi'} \lambda \cdot K(d)(\pi, \pi') \right\}. \]

The bisimulation metric functional \( B \) can be interpreted as the quantitative analogue to the classic bisimulation game: the goal of the attacker is to maximise the distance between the two players, whereas the defender tries to minimise it. Consider processes \( t \) and \( t' \), and choose a challenge \( t \xrightarrow{a} \pi \) from the attacker \( t \). The defender \( t' \) is now allowed to look for the best possible answer \( t' \xrightarrow{a} \pi_\rho \) to the challenge, where by best answer we mean the one that minimises the (discounted) Kantorovich distance from \( \pi_\rho \). Formally, given \( t \xrightarrow{a} \pi \), for each \( \lambda \cdot K(d)(\pi, \pi') \), we look for \( t' \xrightarrow{a} \pi_\rho \) that maximises the Kantorovich distance between \( \pi \) and the closest distribution reached by the defender. This is formalised by the supremum over the transitions from \( t \). To guarantee symmetry, the roles of the two processes are then swapped, and the worst case is considered, as given by the maximum between the obtained distances.

**Proposition 2 ([46]).** Let \( d : T(\Sigma) \times T(\Sigma) \to [0,1] \) be a 1-bounded pseudometric. Then \( B(d) \leq d \) if and only if \( d \) is a bisimulation metric.

The bisimilarity metric \( b^\lambda \), i.e. the smallest bisimulation metric, can then be characterised as the least fixed point of the bisimulation metric functional.

**Proposition 3 ([46]).** The bisimilarity metric \( b^\lambda \) is the least fixed point of \( B \), i.e., \( B(b^\lambda) = b^\lambda \).

Henceforth, in the examples, we will always tacitly assume the fix-point characterisation of the bisimilarity metric, as it allows for a more intuitive evaluation of the distances.
5.5. Comparing approximate bisimulations and metrics

The following example shows the incomparability of \( \varepsilon \)-bisimulations with the bisimilarity metric. This is mainly due to the fact that \( \varepsilon \)-bisimulations are only as good as the worst approximation, while bisimulation metrics always take into account all the differences encountered along the execution path.

Example 14. Consider processes \( t \) and \( u \) given by the first two NPTSs in Fig. 4. Since, \( t_2 \xrightarrow{a} u_2 \xrightarrow{a} \), processes \( t_3 \) and \( u_4 \) cannot be related by any \( \varepsilon \)-bisimulation. On the other hand, \( t_1 \) and \( u_1 \) can be related by all \( \varepsilon \)-bisimulations for all \( \varepsilon \leq 1 \). We infer that \( t_1 \) and \( u_1 \), and thus \( t \) and \( u \), are 0.5-bisimilar and cannot be \( \varepsilon \)-bisimilar for any \( \varepsilon < 0.5 \). However, the bisimilarity distance between them is given by

\[
b^4(t, u) \leq \lambda \cdot \left(0.1 \cdot b^4(t_1, u_1) + 0.9 \cdot b^4(t_2, u_2)\right)
\]

\[
\leq \lambda \cdot \left(0.1 \cdot \lambda \cdot (0.5 \cdot b^4(t_3, u_3) + 0.5 \cdot b^4(t_4, u_4)) + 0.9 \cdot 0\right)
\]

\[
= \lambda^2 \cdot 0.1 \cdot (0.5 \cdot 1 + 0.5 \cdot 0)
\]

\[
= \lambda^2 \cdot 0.05.
\]

Hence, even in the no-discount case \( \lambda = 1 \), processes \( t, u \) are at a bisimilarity distance that is smaller than their approximate bisimulation distance. In this case, the disparity is given by the fact that in the evaluation of \( b^4 \) the distance between the distributions \( \pi_1 = 0.5\delta_{t_1} + 0.5\delta_{t_2} \) and \( \pi_2 = 0.5\delta_{u_1} + 0.5\delta_{u_4} \) is weighted by the probability assigned to the pair \( (t_1, u_1) \) by the matching. Hence, the evaluation of the metrics allows us to take into account the fact that these two distributions have a small probability to be reached, and thus their difference has a lesser impact on the overall distance between \( t \) and \( u \).

Consider now process \( t \) and process \( v \) given by the third NPTS in Fig. 4. A 0.1-bisimulation with all pairs \( (t_i, v_i) \), with \( i = 1 \ldots 4 \), and \( (t, v) \) can be given. On the other hand, we have that

\[
b^4(t, v) = \lambda \cdot \left(0.1 \cdot b^4(t_1, v_1) + 0.8 \cdot b^4(t_2, v_2) + 0.1 \cdot b^4(t_3, v_3)\right)
\]

\[
\leq \lambda \cdot \left(0.1 \cdot \lambda \cdot (0.4 \cdot b^4(t_3, v_3) + 0.5 \cdot b^4(t_4, v_4) + 0.1 \cdot b^4(t_1, v_1)) + 0.1\right)
\]

\[
= \lambda \cdot 0.1 + \lambda^2 \cdot (0.1 \cdot (0.4 \cdot 0 + 0.5 \cdot 0 + 0.1 \cdot 1))
\]

\[
= \lambda \cdot 0.1 + \lambda^2 \cdot 0.01
\]

where we have used equalities because it is straightforward to prove that the ones considered are the optimal matchings. Hence, for \( \lambda = 1 \) the bisimilarity distance between \( t \) and \( v \) is greater than their approximated bisimulation distance. In this case, the disparity is given by the fact that the bisimilarity metric sums up the (weighted) difference between the distributions \( \pi_1 = 0.4 \cdot \delta_{t_3} + 0.5 \cdot \delta_{t_4} \) to the difference between distributions \( 0.1 \cdot \delta_{t_1} + 0.9 \cdot \delta_{t_2} \) and \( 0.2 \cdot \delta_{v_1} + 0.8 \cdot \delta_{v_2} \). Conversely, the approximated bisimulation distance considers these differences independently one from another.

6. Compositional

In this section we introduce the properties that favour compositional reasoning in the probabilistic setting.

For the equivalence approach to semantics, we follow the same approach as of classic process algebras: compositionality consists in requiring that equivalent systems are inter-replaceable. Hence, if a system \( S \) contains as a sub-component a system \( S_1 \), then the behaviour of \( S \) should not change if \( S_1 \) is replaced by an equivalent system \( S_2 \). If we move to behavioural distances, then the requirement is that systems are approximately inter-replaceable: the behaviour of \( S \) should have only a smooth and limited change if \( S_1 \) is replaced by a system \( S_2 \) having a behaviour close to that of \( S_1 \).
In the case of behavioural equivalences, the intuition of inter-replaceability for processes is formalised by requiring that an equivalence relation satisfies the property of congruence.

**Definition 27 (Congruence).** We say that an equivalence relation \( R \subseteq T(\Sigma) \times T(\Sigma) \) is a congruence for an n-ary operator \( f \in \Sigma \) if whenever \( t_i R t'_i, \ldots , t_n R t'_n \), it holds that \( f(t_1, \ldots , t_n) R f(t'_1, \ldots , t'_n) \).

In order to deal with behavioural distances, we need a quantitative analogue to Definition 27, expressing that a small variation in the behaviour of the components leads to a bounded small variation in the behaviour of the composed processes. More precisely, the distances between the components and the distances between the composed processes should be related by some real values function enjoying some property of continuity. Technically, this would allow us to establish that, given any operator \( f \), if we fix a non-zero distance \( \epsilon \), understood as the admissible tolerance from the operational behaviour of a given composed process \( f(t_1, \ldots , t_n) \), then there are non-zero distances \( \epsilon_i \) such that the distance between the composed processes \( f(t_1, \ldots , t_n) \) and \( f(t'_1, \ldots , t'_n) \) is at most \( \epsilon \) whenever the component \( t'_i \) is in distance at most \( \epsilon_i \) from \( t_i \). The intuitions above are formalised by the notion of *modulus of continuity* for \( f \) and the considered distance.

**Definition 28 (Modulus of continuity, [61,62,64–66]).** A mapping \( m : [0,1]^n \rightarrow [0,1] \) is a modulus of continuity of an n-ary operator \( f \) with respect to a pseudometric \( d \) on \( T(\Sigma) \) if:

- \( d(f(t_1, \ldots , t_n), f(t'_1, \ldots , t'_n)) \leq m(d(t_1, t'_1), \ldots , d(t_n, t'_n)) \) for all \( t_1, t'_1, \ldots , t_n, t'_n \in T(\Sigma) \),
- \( m(0, \ldots , 0) = 0 \),
- \( \lim_{\epsilon \to 0} m(\epsilon, \ldots , \epsilon) = 0 \) (namely \( m \) is continuous at \( (0, \ldots , 0) \)).

The notion of modulus of continuity allows us to introduce the notion of uniform continuity for an operator with respect to a pseudometric.

**Definition 29 (Uniformly continuous operator, [61,62,64–66]).** An operator \( f \) is uniformly continuous with respect to a pseudometric \( d \) over \( T(\Sigma) \) if it admits a modulus of continuity with respect to \( d \). In particular:

- \( f \) is non-extensive if the modulus of continuity is \( m(\epsilon_1, \ldots , \epsilon_n) = \max_{i=1}^{n} \epsilon_i \),
- \( f \) is non-expansive if the modulus of continuity is \( m(\epsilon_1, \ldots , \epsilon_n) = \sum_{i=1}^{n} \epsilon_i \),
- \( f \) is \( L \)-Lipschitz continuous for some \( L \geq 1 \) if the modulus of continuity is \( m(\epsilon_1, \ldots , \epsilon_n) = L \cdot \sum_{i=1}^{n} \epsilon_i \).

The reason why Definition 29 uses the uniform version of continuity, instead of the more general notion of continuity (technically, \( m \) depends only on the distance between \( t_i \) and \( t'_i \) and is independent of the concrete processes \( t_i \) and \( t'_i \)), is that it allows for universal compositional guarantees.

Clearly, 1-Lipschitz continuity coincides with non-expansiveness. The most studied notions of compositional behaviour of distances are non-extensiveness and non-expansiveness (see, e.g., [9,46,50,51,63,82,107,112]). However, in [62,66] it is argued that they do not cover recursive operators, for which the compositional reasoning requires Lipschitz continuity.

We notice that whenever an operator \( f \) is uniformly continuous with respect to a 1-bounded pseudometric \( d \), then the fact that the modulus of continuity maps \((0, \ldots , 0)\) to 0 ensures that the kernel of \( d \) is a congruence with respect to \( f \). In general, the opposite implication is not true.

**6.1. The formats**

The crux of this survey consists in presenting the rule formats that have been proposed in the literature for a systematic verification of the compositional properties introduced above.

In Section 7, we discuss the formats proposed [87] to prove the congruence property for generative and reactive bisimilarity. Then, Section 8 deals with the congruence formats over NPTSs for (bi)similarity [17,32,33,40,42], and for (rooted) branching bisimilarity [36,91].

In Section 10, we recall the formats from [109,110] ensuring that approximate generative and reactive bisimulations are non-expansive. The formats for the continuity properties of \( \epsilon \)-bisimulations, introduced in [63], are then presented in Section 11.

Finally, in Section 13, we discuss the continuity formats proposed in [34] for the bisimilarity metric.

**Part 2. Formats for equivalences**

**7. Congruence formats for generative, and reactive bisimulations**

In this section, we recall the formats ensuring the congruence properties for generative and reactive bisimulations. The results of the section stem from [17,87], and are listed in Table 6.
Table 6
Summary of the formats presented in Section 7.

<table>
<thead>
<tr>
<th>Format definition</th>
<th>Compositionality property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generative bisimulation</td>
<td>GTSS</td>
</tr>
<tr>
<td>Definition 14</td>
<td>Theorem 1 [87]</td>
</tr>
<tr>
<td>Reactive bisimulation</td>
<td>RTSS</td>
</tr>
<tr>
<td>Definition 14</td>
<td>Theorem 2 [87]</td>
</tr>
</tbody>
</table>

7.1. Generative bisimulation as a congruence

We start by showing that the pattern of the generative rules (Definition 11) cannot be extended in any trivial way without risking to compromise the congruence property of generative bisimilarity.

Firstly, we argue that one cannot replace the variable \( y \), appearing in the target of a positive premise \( x \frac{a,y}{z} y \), with a more structured term, or allowing a source variable to appear more than once in the source of the rule, or allowing more than one operator to appear in the source of the rule. These requirements are not related to the use of probability, and were already known in the theory of bisimilarity for classic process algebras. We discuss them in the following example.

Example 15. Let us define constants \( k \) and \( k' \), unary operators \( f, h_1 \) and \( h_2 \), and binary operator \( g \) by means of the following rules:

\[
\begin{align*}
\frac{}{ a \frac{1}{1}} k & \quad \frac{a \frac{1}{1}}{ k'} \quad \frac{x \frac{a,y}{z}}{ f(x) \frac{a}{a}} k \quad \frac{a \frac{1}{1}}{ g(x,x) \frac{a}{a} nil} \quad \frac{a \frac{1}{1}}{ h_1(h_2(x)) \frac{a}{a} nil} \\
\end{align*}
\]

Only the rules for \( k \) and \( k' \) are generative rules; the others violate Definition 11: in the rule for \( f \) the target of the premise is not a variable, in the rule for \( g \) the source variable \( x \) appears more than once in the source of the rule, in the last rule the source contains more than one operator. We can easily show that generative bisimulation is not a congruence with respect to these three operators \( f, g \) and \( h_1 \). In detail, we have that \( k \overset{}{\sim}_G k' \) but \( f(k) \not\overset{}{\sim}_G f(k') \), since \( f(k) \overset{a}{\rightarrow} f(k') \overset{a}{\rightarrow} \), and \( g(k,k) \not\overset{}{\sim}_G g(k,k') \), since \( g(k,k) \overset{a}{\rightarrow} g(k,k') \overset{a}{\rightarrow} \). Finally, \( nil \overset{}{\sim}_G h_1(nil) \), since there is no rule for \( h_2 \) thus implying that \( h_2(t) \) has no transitions for all processes \( t \), but \( h_1(nil) \overset{a}{\rightarrow} h_1(h_2(nil)) \) since \( h_1(nil) \overset{a}{\rightarrow} \) and \( h_1(h_2(nil)) \overset{a}{\rightarrow} \).

Then, the following example shows that in generative rules, the probability weights appearing in premises must be variables that can only take the value 0 or 1. Only in this case can generative bisimulation be extended to a congruence, as long as the two rules have the same probability weight.

Example 16. Let us define an operator \( f \) by means of the following rule:

\[
\frac{a \frac{1}{1}}{ x \frac{1}{1} \rightarrow f(y)}
\]

The premise of the rule, which is not allowed in Definition 11, can be instantiated only with a transition with the probability weight 1. Consider the processes \( a.nil \) and \( a.nil +_q a.nil \), with \( q \) an arbitrary value in \((0,1)\). We have \( a.nil \overset{}{\sim}_G a.nil +_q a.nil \) but \( f(a.nil) \not\overset{}{\sim}_G f(a.nil +_q a.nil) \), since \( f(a.nil) \overset{a}{\rightarrow} but f(a.nil +_q a.nil) \overset{a}{\rightarrow} \).

In order to ensure semistochasticity of processes, we need some restrictions on the set of the generative rules forming a TSS, namely we need a specification format. Specifically, one cannot consider an arbitrary set of generative rules respecting Definition 11: For instance, specifying a constant \( k \) by means of the following two rules would not be legal, since the overall probability mass of the transitions of process \( k \) is 2.

\[
\begin{align*}
\frac{a \frac{1}{1}}{1 \rightarrow nil} & \quad \frac{a \frac{1}{1}}{2 \rightarrow nil} \\
\end{align*}
\]

For a set of generative rules \( R \), let \( R' \) denote the subset of the rules in \( R \) having \( f(x_1,\ldots,x_n) \) as source.

Definition 30 (GTSS, [87]). A generative transition system specification, or GTSS for short, is formed by a set \( R \) of generative rules as in Definition 11 such that, for each operator \( f \in \Sigma \), the set \( R' \) of the rules in \( R \) for \( f \) is partitioned into (possibly infinite, but countable) sets \( R'_1,\ldots,R'_n,\ldots \), such that:

1. Given two sets \( R'_a \neq R'_b \), there is an argument \( l \) of operator \( f \) such that:
(a) all rules in one of those two sets have a negative premise \( x_j \overset{C_j}{\rightarrow} \), and
(b) each rule in the other set has an unquantified premise \( x_j \overset{B_i}{\rightarrow} \) for a set \( B_i \) such that \( C_j \cap B_i \neq \emptyset \), or an active premise
\[ x_j \overset{a_i}{\rightarrow}, \quad y_i \]
for an action \( a_i \) such that \( a_i \in C_i \).

2. for each set \( R'_i \), it holds that:

(a) all rules in \( R'_i \) have the same sets of active variables, unneeded premises, unquantified premises and negative premises;
(b) given a set of actions \( \{ a_i | i \in I \} \) such that \( \{ x_j | i \in I \} \) are the active variables of the rules in \( R'_i \), if the rules in \( R'_i \) have no unneeded premise \( x_j \overset{A_i}{\rightarrow} \) with \( i \in I \) and \( a_i \in A_i \), then there is at least one rule in \( R'_i \) with active premises \( \{ x_j \overset{a_i}{\rightarrow}, y_i | i \in I \} \);
(c) if \( r_1, \ldots, r_m \), are the rules in \( R'_i \) having the same set of active premises, then their weights satisfy \( w_{r_1} + \cdots + w_{r_m} + \cdots = 1 \).

Clause 1 in Definition 30 ensures that the transitions of a process \( f(t_1, \ldots, t_n) \) cannot be derived by applying rules from two different sets \( R'_i \) and \( R'_j \). Then, clause 2 ensures that if some transitions from \( f(t_1, \ldots, t_n) \) are derivable from a set \( R'_i \), then their probability weights sum up to 1.

Example 17. The set of all rules in Table 3 and Table 4 give a GTSS. As already commented in Example 12, the rules for finite iteration \( R^{+1} \) are partitioned into two sets, one characterised by the unquantified premise for \( \sqrt{\ } \), the other characterised by the negative premise for the same label. The sets \( R^{+3} \) and \( R^{+4} \) are partitioned into three subsets: one has active premises for both arguments, one has the negative premise for the first argument, the other has the negative premise for the second argument. The set \( R^0 \) is partitioned in four subsets, discriminated by the unquantified or negative premise for \( \sqrt{\ } \) for the two arguments. The set \( R^{+5} \) is partitioned in two sets, one containing rules with the unquantified premise for \( a \), the other containing rules with the negative premise for the same label. □

The following example shows that extending generative rules by allowing terms more general than variables in the source of premises, in the style of the so called týt-rules in [75], would not be trivial at all.

Example 18. Let us define the unary operators \( f \) and \( g \) by means of the following rules,
\[ r_{a,1} = \frac{x \overset{a}{\rightarrow}, y}{f(x)} \quad r_{a,2} = \frac{g(x) \overset{a}{\rightarrow}, y}{f(x)} \quad r_{a,3} = \frac{x \overset{a}{\rightarrow}, y}{g(x) \overset{a}{\rightarrow}} \]
where we have a rule \( r_{a,1} \) and a rule \( r_{a,2} \) for each action \( a \in A \) and a rule \( r_{a,3} \) for each \( a \neq b \). Notice that \( w_1, w_2 \in (0, 1) \) are the weights for \( r_{a,1} \) and \( r_{a,2} \). Assume two processes \( t_1 \) and \( t_2 \) and that both these processes have transitions with overall probability 1. Assume that \( t_1 \overset{b}{\rightarrow} t_2 \). Since \( t_1 \overset{b}{\rightarrow} t_2 \), then \( g(t_1) \) has no transition and the transitions of \( f(t_1) \) can be derived by applying the rules \( r_{a,1} \) but not the rules \( r_{a,2} \). Therefore, \( f(t_1) \) is semistochastic if and only if \( w_1 = 1 \). Since \( t_2 \overset{b}{\rightarrow} t_2 \), then there are transitions from \( g(t_2) \) thus implying that the transitions of \( f(t_2) \) can be derived by applying both the rules \( r_{a,1} \) and \( r_{a,2} \), thus implying that \( f(t_2) \) is semistochastic only if \( w_1 + w_2 = 1 \). Summarising, in order to have that both \( f(t_1) \) and \( f(t_2) \) are semistochastic, we need that \( w_1 = 1 \) and \( w_1 + w_2 = 1 \) for \( w_1, w_2 \in (0, 1) \), which is not possible. In order to fix such an issue, we should: (i) opt for \( w_1 \) with \( w_1 + w_2 = 1 \), (ii) replace each \( r_{a,1} \) with two rules, say \( r_{a,1}^{up} \) and \( r_{a,1}^{up} \), (iii) add negative premises \( NP \) to \( r_{a,1}^{up} \) and unquantified premises \( UP \) to \( r_{a,1}^{up} \) in order to ensure that \( r_{a,1}^{up} \) can be applied to derive transitions of a process \( f(t) \) only if \( g(t) \) cannot move and \( r_{a,1}^{up} \) can be applied to derive transitions of a process \( f(t) \) only if \( g(t) \) can move, (iv) assigning weight 1 to \( r_{a,1}^{up} \) and keep the weight \( w_1 \) for \( r_{a,1}^{up} \). Namely, we should replace \( r_{a,1} \) with the rules,
\[ r_{a,1}^{up} = \frac{x \overset{a}{\rightarrow}, y}{f(x)} \quad r_{a,1}^{up} = \frac{x \overset{a}{\rightarrow}, y}{f(x)} \]
with \( NP \) negative premises formalising that \( g(x) \) cannot move and \( UP \) unquantified premises formalising that \( g(x) \) moves. This is doable, but it would entail a complex rewriting of clause 1 in Definition 30. □

From the results in [17,87] one can infer that generative bisimilarity over processes specified by generative rules is a congruence. In [87] it is shown also that the constraints imposed by GTSSs ensure that processes are semistochastic. Those results were later extended to generative rules admitting double testing.

**Theorem 1 ([87]).** Assume the GPTS induced by a GTSS. Then, (i) all processes are semistochastic, and (ii) the generative bisimilarity over the processes is a congruence.
7.2. Reactive bisimulation as a congruence

The GPTS derived from the rules in Examples 15 and 16 is also a RPTS. Therefore, the constraints in the pattern of the rules discussed in those examples are mandatory also in case of reactive rules.

In order to ensure that Definition 7 is respected, namely that all processes enjoy the property of reactivity, we need some restrictions on sets of rules forming a TSS. For instance, defining a constant $k$ as in Equation (1), with the label $b$ of the second rule replaced by the $a$ of the first rule, would not be legal. For a set of reactive rules $R$, let $R^{f,a}$ denote the subset of the rules in $R$ having $f(x_1, \ldots, x_n)$ as source and $a$ as action.

**Definition 31** (RTSS, [87]). A reactive transition system specification, or RTSS for short, is formed by a set $R$ of reactive rules such that, for each operation $f \in \Sigma$ and action $a \in A$, the set $R^{f,a}$ is partitioned into (possibly infinite, but countable) sets $R^{f,a}_1, \ldots, R^{f,a}_n, \ldots$ such that:

1. Given two sets $R^{f,a}_u \neq R^{f,a}_v$, there is an argument $l$ of operator $f$ such that:
   
   (a) all rules in one of those two sets have a negative premise $x_j \rightarrow C_j$, and
   (b) each rule in the other set has an unquantified premise $x_j \rightarrow B_j$ for a set $B_j$ such that $C_j \cap B_j \neq \emptyset$, or an active premise $x_j \rightarrow a_i, y_i$ for an action $a_i$ such that $a_i \in C_j$.

2. For each set $R^{f,a}_u$, it holds that:
   
   (a) all rules in $R^{f,a}_u$ have the same set of unquantified premises and negative premises;
   (b) the sum of the weights of the rules in $R^{f,a}_u$ is 1;
   (c) given any two rules $r_1$ and $r_2$ in $R^{f,a}_u$, then, for each active premise $x_i \rightarrow a_i, y_i$ in $r_1$, there is either an active premises $x_i \rightarrow a_i, y_i$ in $r_2$, or an unquantified premise $x_i \rightarrow B_j$ in $r_2$ with $a_i \in B_j$.

Clause 1 in Definition 31 ensures that the transitions of a process $f(t_1, \ldots, t_n)$ labelled $a$ cannot be derived by rules in two different sets of rules $R^{f,a}_u$ and $R^{f,a}_v$. Then, clause 2 ensures that the probability of the $a$-labelled moves of $f(t_1, \ldots, t_n)$ that are derivable from any set $R^{f,a}_u$ (if there are any) sum up to 1.

**Example 19.** The set of the rules in Table 3 and Table 5 give a RTSS.  

Notice that clause 1 in Definition 30 guarantees mutual exclusion between any pair of sets of rules $R^{f}_u$ and $R^{f}_v$ for $f$, whereas clause 1 in Definition 31 guarantees mutual exclusion between $R^{f,a}_u$ and $R^{f,a}_v$ only if $a = b$. The reason is that the probability of the moves of a process labelled $a$ and the probability of the moves of the same process labelled $b$, with $a \neq b$, are related only in the generative case.

The following result was proved in [87] also for reactive rules admitting double testing.

**Theorem 2** ([87]). Assume the RPTS induced by a RTSS. Then, (i) for each process, the probability of its outgoing transitions with a given label $a$, if any, sum up to 1, and (ii) the reactive bisimilarity over the processes is a congruence.

8. Congruence formats over NPTSs

In this section, we consider the model of nondeterministic and probabilistic processes and recall the formats ensuring the congruence property for strong (bi)similarities and branching bisimilarity, developed in [17,33,40,42] and [36,91], respectively, and are summarised in Table 7.

8.1. Strong (bi)simulations as a congruence

The (pre)congruence formats for bisimilarity, ready similarity, and similarity over NPTSs have been obtained via a probabilistic version of the GSOS format of [27]: the Probabilistic GSOS (PGSOS) format.

**Definition 32** (PGSOS TSS, [16,17,39,40,42]). A PSOS rule as in Definition 10 is a Probabilistic GSOS (PGSOS) rule if:

1. all process variables $x_1, \ldots, x_n$ are pairwise different, and
2. all distribution variables $\mu_{i,k}$ for $i \in I$ and $k \in K_i$ are pairwise different.

A PGSOS rule is called positive if it contains only positive premises. A PSOS TSS $(\Sigma, A, R)$ is a (positive) PGSOS TSS if all the rules in $R$ are (positive) PGSOS rules.
We notice that the rules specifying most of operators in [6,7,17,42,47,49,80], included those in Table 2, are in PGSOS format. This witnesses the relevance of the following result.

**Theorem 3.** Assume a PGSOS TSS $T$. Then, bisimilarity is a congruence [17,40,42] and ready similarity is a precongruence [33]. Moreover, if $T$ is positive, also similarity is a precongruence [33].

We now provide a simple example showing that by relaxing the syntactical constraints in Definition 32, the congruence property can no longer be guaranteed. It is also shown that precongruence of similarity is broken by negative premises. This example is obtained as a straightforward adaptation to the PSOS setting of classic counterexample from the literature on nondeterministic processes (see, e.g., [75, Section 5]).

**Example 20.** Let us consider the operators $f,g,h$ defined by the following PSOS rules:

\[
\begin{align*}
\frac{x_1 \xrightarrow{a} \mu_1}{f(x_1,x_1) \xrightarrow{a} \mu_1} & \quad \frac{x_1 \xrightarrow{a} \mu_1}{g(x_1,x_2) \xrightarrow{a} \mu_1} & \quad \frac{x_1 \xrightarrow{a} \mu_1}{h(x_1) \xrightarrow{a} \mu_1}
\end{align*}
\]

Notice that the rule for $f$ violates item 1 in Definition 32 and the rule for $g$ violates item 2. We proceed to show that bisimilarity is not a congruence for $f$ and $g$. Consider processes $t = a.nil +_0.5 a.nil$ and $t' = a.nil +_0.5 a.(a.nil +_0.5 nil)$, which should be equated by all reasonable notions of equivalence. As expected, we have $t \sim t'$. However, $f(t,t') \not\sim f(t',t')$ and $g(t,t') \not\sim g(t',t')$. Indeed, if we name by $\pi$ the distribution $0.5\delta_{a.nil} + 0.5\delta_{b.nil}$, we get $f(t,t') \xrightarrow{\pi} \pi$ and $f(t',t') \xrightarrow{\pi}$, since the latter process has no transitions. Similarly, $g(t,t') \xrightarrow{\pi}$ and $g(t',t') \xrightarrow{\pi}$, as the rule for $g$ requires the distribution terms in the premises to be syntactically indistinguishable. Moreover, we show that the negative premise in the rule for $h$ breaks similarity. For process $t +_0.5 h.nil$ we have $t \subseteq t +_0.5 h.nil$ but $h(t) \not\subseteq h(t +_0.5 h.nil)$, since $h(t) \xrightarrow{\pi}$ but $h(t +_0.5 h.nil) \xrightarrow{\pi}$. 

Actually, in [40,42] the congruence result was proved for rules with a pattern more general than the PGSOS presented in Definition 32, namely the $nt\mu\theta/nt\mu x\theta$ rules, which can be thought of as the probabilistic counterpart of the $nt\psi/nt\psi\theta$ pattern from [74]. In detail, the symbols in $nt\mu\theta/nt\mu x\theta$ have the following meaning: (i) “n”: negative premises are allowed, (ii) “t$\mu$”: premises have the form $r \xrightarrow{\mu} \mu$ for a generic term $t$, (iii) “$t\theta$”: the conclusion has the form $f(x_1,\ldots,x_n) \xrightarrow{\theta} \theta$, and (iv) “x$\theta$”: the conclusion has the form $x \xrightarrow{\theta} \theta$. We recall here a restricted version of those rules, called *simple nt$\mu\theta$/nt$\mu x\theta$ rules* in [63].

**Definition 33.** [Simple nt$\mu\theta$/nt$\mu x\theta$-TSS, [40,42,63]] The simple nt$\mu\theta$ rules and the simple nt$\mu x\theta$ rules are, respectively, of the form

\[
\begin{align*}
\frac{[t_i \xrightarrow{a} \mu_i | i \in I]}{f(x_1,\ldots,x_n) \xrightarrow{\theta}} & \quad \frac{[t_j \xrightarrow{a} j | j \in J]}{g(x_1,\ldots,x_m) \xrightarrow{\theta}} & \quad \frac{[t_i \xrightarrow{a} \mu_i | i \in I]}{h(x_1) \xrightarrow{\theta}}
\end{align*}
\]

with $I,J$ countable sets of indexes, $t_i,t_j \in T(\Sigma,\nu_p)$, $\alpha_i,\alpha_j, \alpha \in A$, $\mu_i \in V_d$, $f \in \Sigma$, $x_1,\ldots,x_n \in V_p$, $\theta \in DT(\Sigma,\nu_p,\nu_d)$, $x \in V_p$ and the constraints 1-2 in Definition 32 hold.

The rules in [40,42] admit premises expressing that, intuitively, some variables appearing in a term $t_k$ can be in the support of a distribution variable $\mu_k$, which allows for inspecting several subsequent transitions of a process argument.

### 8.2. Branching bisimulation

Two proposals of congruence formats for (rooted) branching bisimilarity can be found in [36,91]. Although the sets of syntactical constraints characterising the formats are similar, the two approaches differ on the construction of the PSOS rules. Roughly, in [91]...
Example 21. Both arguments of \( r, r' \) should be marked as N-liquid, since in a given process \( r, r' \) both \( r \) and \( r' \) are able to start running immediately. Then, the first argument is \( \Lambda \)-liquid and the second \( \Lambda \)-frozen, since only \( r \) may have already performed some transitions. The argument of \( r, r' \), \( r'' \), \( \Lambda r \) and \( \Lambda r'' \) is \( \Lambda \)-frozen, as it is not currently running, and \( N \)-liquid, since it can start running right away. Both arguments of operator \( r + r' \) are \( \Lambda \)-frozen, since in a given process \( r + r' \) both \( r \) and \( r' \) cannot have already performed any transition, and \( \Lambda \)-liquid, since both \( r \) and \( r' \) are able to start running immediately. The labelling of the arguments of \( r + r' \) is similar. Then, the arguments \( i \) of a \( \bigotimes_{i \in I} [p_i] \) neither have preformed any action yet, nor they can start their execution immediately (since first action \( a \) has to be performed). Hence, they are marked \( \Lambda \)-frozen and \( N \)-frozen. Conversely, both arguments of operators \( r \parallel r' \), \( r \parallel t \), \( r \parallel t' \), \( t \parallel t' \), \( t \parallel \Lambda t \), \( t \parallel \Lambda t' \), \( t \parallel \Lambda t'' \), \( t \parallel \Lambda t''' \), \( \Lambda t \parallel \Lambda t' \), \( \Lambda t \parallel \Lambda t'' \), \( \Lambda t \parallel \Lambda t''' \) are \( \Lambda \)-liquid and \( N \)-liquid: in fact, they may have already performed some computation steps, and they can also proceed with their execution at once. Finally, the argument of \( \{f\} \setminus t \), \( n_L, n_R(t) \) is \( \Lambda \)-liquid and \( N \)-liquid. \( \blacksquare \)

Informally, in the construction of the format, we can use \( \Lambda \) to distinguish the arguments of an operator \( f \) that cannot be tested in the premises of the rules for \( f \) (i.e., those labelled as \( N \)-frozen), from the ones that may be tested (i.e., those labelled as \( \Lambda \)-liquid). Then, we can use the marking \( \Lambda \) over the \( \Lambda \)-liquid arguments to distinguish those for which the root condition must hold (i.e., the \( \Lambda \)-frozen ones), from those that are not required to always satisfy it (i.e., the \( \Lambda \)-liquid ones).

From this characterisation, albeit intuitive, we can infer that the testing for \( r \)-moves of arguments that are marked as both \( \Lambda \)-liquid and \( N \)-liquid requires care. In fact, since the root condition is not required to hold for these arguments, we need to ensure that the presence (or absence) of a \( r \)-transition by them does not entail a particular behaviour of the process in which they occur.

Example 22. Consider processes \( a.\delta \) and \( a.\delta' \), where \( t = a.\delta || b.\delta_n \) and \( r' = a.\delta || t.\delta \), for \( t'' = b.\delta_n \). Clearly, \( t \approx_b r' \) and \( \forall \delta \in a.\delta_n \), \( a.\delta \approx_a a.\delta_n \). Assume now an operator \( f \) with the following rules:

\[
\begin{align*}
\text{x} & \xrightarrow{a} \mu \\
 f(x) & \xrightarrow{a} f(\mu) \quad f(x) & \xrightarrow{r} \delta_n
\end{align*}
\]

We have that \( f(t) \approx_b f(t') \) (and, thus, \( f(a.\delta) \approx_b f(a.\delta') \) since \( f(t) \approx_t f(t') \)) and \( f(t) \approx_t f(t') \). \( \blacksquare \)

For this reason, we require \( r \)-moves by \( (\Lambda \cap \Lambda \cap \Lambda) \)-liquid arguments to be tested only in patience rules, namely rules allowing a process to mimic the \( r \)-moves of its arguments without modifying its structure.

Definition 34 (Patience rule). A patience rule for argument \( i \) of \( f \) is a PGSOS rule of the form:

\[
\begin{align*}
\text{x_1, \ldots, x_n} & \xrightarrow{r} f(\delta_{x_1}, \ldots, \delta_{x_n}, \mu, \delta_{x_{i+1}}, \ldots, \delta_{x_n})
\end{align*}
\]

In Definition 35 below, we formalise the “liquid”/“frozen” marking of the arguments of operators, and show how it can be extended to the variables occurring in (distribution) terms.

Definition 35 (Liquid and frozen arguments of operators, [26,36,59]). Let \( \Gamma \in \{\Lambda, \Lambda \} \) be an unary predicate on \( \{f(i) \mid f \in \sum, 1 \leq i \leq n\} \). If \( \Gamma(f, i) \) then argument \( i \) of \( f \) is labelled as \( \Gamma \)-liquid, otherwise it is \( \Gamma \)-frozen. Let \( t \) be a term, \( \theta \) a distribution term and \( x \in \mathcal{V}_p \) and \( \mu \in \mathcal{V}_d \) variables. Then:

1. An occurrence of \( x \) in \( t \) is \( \Gamma \)-liquid if: (i) either \( t = x \); or (ii) or \( t = f(t_1, \ldots, t_n) \), that occurrence of \( x \) is \( \Gamma \)-liquid in \( t_i \); and \( i \) is a \( \Gamma \)-liquid argument of \( f \).
2. An occurrence of \( x \) in \( \theta \) is \( \Gamma \)-liquid if: (i) \( \theta = \delta \) and \( x \) occurs \( \Gamma \)-liquid in \( t \); or (ii) \( \theta = f(\theta_1, \ldots, \theta_n) \), that occurrence of \( x \) is \( \Gamma \)-liquid in \( \theta_1 \); and \( i \) is a \( \Gamma \)-liquid argument of \( f \), or (iii) \( \theta = \sum_{i \in I} \theta_i \) and the occurrence of \( x \) is \( \Gamma \)-liquid in some \( \theta_i \).
3. An occurrence of \( \mu \) in \( \theta \) is \( \Gamma \)-liquid if: (i) \( \theta = \mu \); or (ii) \( \theta = f(\theta_1, \ldots, \theta_n) \) and the occurrence of \( \mu \) is \( \Gamma \)-liquid in \( \theta_1 \); for a \( \Gamma \)-liquid argument \( i \) of \( f \), or (iii) \( \theta = \sum_{i \in I} \theta_i \) and the occurrence of \( \mu \) is \( \Gamma \)-liquid in some \( \theta_i \).

Then, occurrences of \( x \) and \( \mu \) that are not \( \Gamma \)-liquid in \( t \) or \( \theta \) are \( \Gamma \)-frozen. We say that a predicate \( \Gamma \) is universal if it holds for all arguments of all operators in the signature.
We now have all the ingredients necessary to introduce the specification formats for branching bisimilarity and its rooted version, i.e., respectively, the PBB-safe format and the PRBB-safe format. These can be thought of either as extensions of the GSOS format to capture the considered weak semantics, or as probabilistic versions of the BB-safe and the RBB-safe formats, respectively, proposed for those equivalences in the non-probabilistic setting [58,59].

Definition 36 (PRBB-safe TSS, [36]). A PGSOS rule $\rho$ is probabilistic rooted branching bisimulation safe (PRBB-safe) with respect to predicates $\mathfrak{N}$ and $\Lambda$ if:

1. Right-hand sides of positive premises occur only $\Lambda$-liquid in the target.
2. If $x$ occurs only $\Lambda$-liquid in the source, then $x$ occurs only $\Lambda$-liquid in $\rho$.
3. If $x$ occurs only $\mathfrak{N}$-frozen in the source, then $x$ does not occur in any premise.
4. If $x$ has exactly one $\mathfrak{N}$-liquid occurrence in source, which is also $\Lambda$-liquid, then: (a) $x$ has at most one occurrence in the positive premises, (b) $x$ does not occur in any negative premise, (c) if $x$ occurs in a premise labelled $\tau$, then $\tau$ must be patient for some argument $i$ of the source.

A TSS is PRBB-safe for some predicates $\mathfrak{N}$ and $\Lambda$, if it has the patience rules for all $(\mathfrak{N} \cap \Lambda)$-liquid arguments and it only contains PRBB-safe rules. We say that it is PBB-safe if, moreover, predicate $\Lambda$ is universal.

In item 1, since a premise $x \overset{a}{\to} \mu$ implies that the processes in the support of any closed instance of $\mu$ are running, it is required that $\mu$ occurs in the target only at $\Lambda$-liquid positions, which are those hosting running processes. Item 2 guarantees that running processes, namely those marked liquid by $\Lambda$, maintain their mark and thus the possibility to run. Item 3 prevents the testing of processes that cannot execute, namely those that are at $\mathfrak{N}$-frozen positions in the source. Item 4 regulates the testing of processes at $(\mathfrak{N} \cap \Lambda)$-liquid positions, namely those for which one has $\pi_0$, but the root condition of $\pi_{ib}$ is not guaranteed. The following example explains clauses 4a and 4b in item 4 in detail, clause 4c was already discussed in Example 22.

Example 23. Two rules for operator $\parallel_p$ in Table 2 violate clause 4b. Consider processes $t = a.b.nil$, $t' = a.c.nil$ and $t'' = \tau.t'$. Clearly, $t' \approx_t t''$. However, $t \parallel_{0.5} t' \parallel_{0.5} \parallel t''$. Indeed, $t \parallel_{0.5} t' \parallel_{0.5} x'$, with $x' = 0.5\delta_{a.nil}t' + 0.5\delta_{a.c.nil}$ and $t \parallel_{0.5} t'' \parallel_{0.5} x''$, with $x'' = 0.5\delta_{a.nil}t''$.

Since $b.nil \parallel_{0.5} t'' b$ and $t \parallel_{0.5} c.nil$ cannot reach any process able to perform $b$ through any sequence of $\tau$-labelled transitions, we infer that $b.nil \parallel_{0.5} t'' b$ and, therefore, $x' \approx x''$ does not hold, thus implying $t \parallel_{0.5} t' \parallel_{0.5} t''$.

Let us consider now an operator $f$ specified by the patience rule and a rule violating clause 4a:

\[
\begin{align*}
\begin{array}{c}
x \overset{a}{\to} \mu_1 \\
\delta_x \overset{\tau}{\to} \delta_y \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
x \overset{b}{\to} \mu_2 \\
\delta_y \overset{\tau}{\to} f(\mu) \\
\end{array}
\end{align*}
\]

Consider the ACP-like processes $a$, $b$ and $\tau$ and, then, $t_1 = (a + b)\omega$ and $t_2 = (a + \tau; (b + \tau))\omega$. We have $t_1 \approx_t t_2$ but $f(t_1) \not\approx f(t_2)$. Indeed, $t_1$ can perform both $a$ and $b$, thus giving $f(t_1) \overset{a}{\to} \Delta$, whereas $t_2$ cannot reach (through any sequence of $\tau$-actions) any process able to make both $a$ and $b$, thus implying that $f(t_2)$ cannot reach (through any sequence of $\tau$-actions) any process able to make $c$.

Theorem 4 ([36,91]). Let $P$ be a PRBB-safe TSS. Then $\approx_{ib}$ is a congruence. Moreover, if $P$ is PBB-safe, then $\approx_{ib}$ is a congruence for all operators defined by $P$.

If we exclude from Table 2 the rules for operators $t \parallel_p t'$ and $\partial_{ib}^p(t)$, then we obtain a PRBB-safe TSS by using the labelling in Example 21. Notably, these two operators break the equivalence. This has already been shown for $\parallel_p$ in Example 23. We can show that this holds also for $\partial_{ib}^p$. For instance, given CCS processes $t = a.b.nil + b.a.nil$ and $t' = \tau.t + b.a.nil$, we get $t \approx_{ib} t'$ but $\partial_{ib}^p(t) \not\approx_{ib} \partial_{ib}^p(t')$, since $\partial_{ib}^p(t) \not\approx_{ib} \partial_{ib}^p(t')$.

9. Formats highlights

In this part we have discussed the formats for probabilistic bisimulation-based relations over the three models.

The GSOS format (Definition 32) for strong (bi)similarity over NPTSs is indeed the most intuitive and simplest of the ones provided. In fact, it is obtained as a straightforward translation of the classic GSOS format onto NPTSs. Similarly, the PRBB-safe format (Definition 36) for rooted branching bisimilarity can be seen as the probabilistic version of the RBB-safe format that ensures the congruence property of the considered equivalence over nondeterministic languages. Specifically, the PRBB-safe format is based on two predicates and a liquid/frozen marking of arguments of operators that, as in the classic case, allows us to distinguish the $\tau$-transitions that should be treated as visible from the silent ones. Moreover, the PBB-safe format for branching bisimilarity is obtained from the PRBB-safe format by requiring the predicate $\Lambda$ to be universal, exactly as the BB-safe format is obtained from the RBB-safe format.

The similarities between the classic and probabilistic formats in these cases are due to the features of NPTSS: in this model, the probabilistic behaviour is expressed entirely by the distribution terms, i.e., the targets of transitions. Indeed, the cases of GPTSs and
RPTs are different, as each transition is assigned a probability weight. Hence, the main technical difficulty in the definition of the GTSS (Definition 30) and the RTSS (Definition 31) formats for strong bisimilarity consists in guaranteeing the semistochasticity, or reactivity, of processes. For this reason, in both formats the rules for an operator are partitioned into mutually exclusive derivation sets such that the sum of the weights of the transitions derivable from each partition is 1. However, in RTSS the rules are partitioned not only according to the source operator, but also considering the label of the conclusion. In fact, in the reactive case the probability of the a-labelled moves of a process sum either to 0 or to 1.

Part 3. Formats for approximated equivalences

10. Non-expansiveness formats for approximate generative, and reactive bisimulations

In this section, we recall the formats from [110] ensuring that both approximate generative bisimulations and approximate reactive bisimulations are non-expansive. See also Table 8 for a summary. Other form of continuity, such as Lipschitz continuity, were not considered in [110]. We observe that in both cases, the generative and the reactive one, any format guaranteeing non-expansiveness of e-bisimulation guarantees also the congruence of bisimulation, since the kernel of an e-bisimulation is a bisimulation. Therefore, the constraints imposed by formats for approximate bisimulations should be at least as strong as those for bisimulations.

10.1. Non-expansiveness of approximate generative bisimulation

We start by recalling the notion of approximate generative bisimulation safe (agb-safe) rule. Essentially, agb-safe rules are generative rules that: (i) admit only a limited form of unquantified and negative premises, (ii) do not admit duplication of variables, i.e., the arguments of operators and their derivatives cannot be duplicated in the target of the rule, and (iii) force the derivatives of source variable to appear in the target.

Definition 37 (agb-safe rule, [110]). A generative rule as in Definition 11 is agb-safe if:

1. unquantified and negatives premises can be labelled only by $A$,
2. for each index $i \in \{1, \ldots, n\} \setminus I$, at most one occurrence of source variable $x_i$ appears in the target $t$,
3. for each index $i \in I$, no occurrence of source variable $x_i$ appears in $t$,
4. for each index $i \in I$ exactly one occurrence of the derivative $y_i$ of the source variable $x_i$ appears in $t$.

We now illustrate that the constraints in Definition 37 are necessary by means of several examples. Negative, or unquantified, premises using a set of actions $A \subseteq \mathcal{A}$ are forbidden (clause 1), since they could distinguish a process argument that does not perform actions in $A$, from an approximately bisimilar process argument that performs actions in $A$ with a small probability. The example below deals with negative premises, an example dealing with unquantified premises could be given in an analogous way.

Example 24 (Negative premises). Assume an action label $a \in A$ and let $f$ be the unary operator having the following rule $r_b$ for all $b \neq a$:

$$
\begin{array}{c}
\frac{X^{b,b}}{x^{b,b}, y} \quad \frac{x^{a}}{f(x)^{b,b}, (t_0, i) y}
\end{array}
$$

For $\epsilon \in (0, 1)$, the processes $t_1 = b$ and $t_2 = b +_{1-\epsilon} a$ are related by a generative $e$-bisimulation, whereas $f(t_1)$ and $f(t_2)$ cannot be related by any $e'$-bisimulation, since $f(t_1)$ can perform $b$ and $f(t_2)$ cannot. ▶

Variable duplication is forbidden: as, in rules, source variables $x_i$ are instantiated by process arguments, and variables $y_j$ by process argument derivatives, their duplication may cause a duplication of the distance between different instances of process arguments, thus breaking non-expansiveness. The following example deals with the duplication of derivatives of source variables. Similar arguments could be given for duplication of source variables, or to show that a source variable and its derivative cannot both appear in the target.
Example 25 (Duplication of variables). Let $f$ be the unary operator having rule $r_f^a$ for all $a \in A$:

$$r_f^a = \frac{a}{f(x)} \xrightarrow{a,x} y \xrightarrow{a,x} \gamma(y,x)$$

Notice that $y$ is duplicated in the rule target, thus violating clause 4. Then, let $g$ be the operator having the following rules $r_g^a$, for all $a \in A$, and $r_{g,b}$, for all $a, b \in A$ with $a \neq b$, where $d$ is an action label in $A$:

$$r_g^a = \frac{x_1}{g(x_1,x_2)} \xrightarrow{a,p_1} \gamma(y_1,x_2)$$

$$r_{g,b} = \frac{x_1}{g(x_1,x_2)} \xrightarrow{a,p_1} \gamma(y_1,x_2)$$

Assume $p = \pi\gamma$ and $\epsilon \in (0,1)$. Consider processes $t_1 = a.(b + \epsilon c)$ and $t_2 = a.(b + \epsilon c)$, and all $\epsilon$-bisimulations $\delta$ for $\delta(t_1)$ and $\delta(t_2)$ are related by no $\epsilon$-bisimulation. In fact, $\delta(t_1)$ and $\delta(t_2)$ can perform $a$ and reach $g(b + \epsilon c, b + \epsilon c)$ and $g(b + \epsilon c, b + \epsilon c)$ for $\epsilon^2$, respectively. Then $g(b + \epsilon c, b + \epsilon c)$ performs $b$ with probability $\epsilon$, whereas $g(b + \epsilon c, b + \epsilon c)$ performs $b$ with probability $\epsilon^2 + \epsilon^2$, thus implying that a $\epsilon$-bisimulation relating these processes would require that $\epsilon \geq \epsilon^2 + \epsilon^2$. □

The derivatives of source variables must appear in the target because, otherwise, it may happen that the distances between different derivatives of processes arguments are summed up.

Example 26 (Derivatives of source variables). Let $f$ be the unary operator having rule $r_f$ for all $a \in A$:

$$r_f = \frac{x}{f(x)} \xrightarrow{a,x} y \xrightarrow{a,x} \gamma(y,x)$$

Variable $y$ does not appear in the rule target, thus violating clause 4. Let $t_1 = (a.b + \epsilon c) + \epsilon_2 (b + \epsilon c)$ and $t_2 = (a.b + \epsilon c) + \epsilon_2 (b + \epsilon c)$ be the processes in Fig. 5. They are $\epsilon$-bisimilar, but $\delta(t_1)$ and $\delta(t_2)$ cannot be related by any $\epsilon$-bisimulation, since they reach nil through $a$ with probability $\epsilon + 2\epsilon_2$ and $\epsilon_2$, respectively, thus implying that $\delta(t_1)$ and $\delta(t_2)$ can be related by a $\epsilon$-bisimulation only if $\epsilon \geq 2\epsilon_2$. The point is that in computing the distance between $\delta(t_1)$ and $\delta(t_2)$ we sum up the distance between $t_1$ and $t_2$ caused by the different probability to reach $b.nil$ and that caused by the different probability to reach $c.nil$. □

As observed in Remark 4, double testing can be added to generative rules without compromising the congruence property for generative bisimulation (Theorem 1). However, we can show that double testing would break non-expansiveness of generative $\epsilon$-bisimulation and, therefore, cannot be admitted in Definition 37. Technically, double testing consist in having rule premises of the form $x_i \xrightarrow{a_i,p_i} \gamma_i x_i$ and $x_i \xrightarrow{a_i,p_i} \gamma_i x_i$ with the same variable $x_i$ in the left side. In order to ensure that the probability of both the mutually exclusive events “$x_i$ performs $a_i$” and “$x_i$ performs $a_i$” contribute to the probability of the derived transition by $\delta(x_i, \ldots, x_n)$, in [87] their probability weights $p_i$ and $p_i$ were summed when computing the probability of the move of $f(x_i, \ldots, x_n)$. However, since this is not the reason for which double testing cannot be admitted, in the example below we let $p_i$ and $p_i$ be composed with an arbitrary function $h : [0,1] \times [0,1] \rightarrow [0,1]$.

Example 27 (Double Testing). Consider two actions $a, b \in A$ and define $f$ as the unary operator having the following rules: $r_f^{a,b}$ (with double testing) for actions $a$ and $b$, and $r_f^c$ for all $c \in A \setminus \{a, b\}$

$$r_f^{a,b} = \frac{x}{f(x)} \xrightarrow{a,b,p_1} \gamma(x_1,x_2)$$

$$r_f^c = \frac{x}{f(x)} \xrightarrow{a,b,p_1} \gamma(x_1,x_2)$$

where $h : [0,1] \times [0,1] \rightarrow [0,1]$ is a suitable function. Then, let $g$ be the binary operator having the following rule $r_g^{a_1,a_2}$ for all pairs of actions $a_1, a_2 \in A$: 29
Fig. 6. Two generative \( e \)-bisimilar processes.

\[
\frac{r_{a_1,a_2}^f}{x_1, x_2} \xrightarrow{\alpha_{1,2}, \beta_{1,2}} (x_1, x_2) \xrightarrow{\alpha_{2,1}} (x_1, x_2) \xrightarrow{\beta_{2,1}} (x_1, x_2) \xrightarrow{\gamma_{1,2}} (x_1, x_2)
\]

where \( k : A \times A \rightarrow A \) is a function injective in \((b, b), (c, c), (b, c), (c, b)\). Consider processes \( t_1 = a.s_1 +_j b.s_1 \) and \( t_2 = a.s_2 +_j b.s_2 \), with \( s_1 = b +_j c \) and \( s_2 = b +_j +_j c \). Processes \( t_1 \) and \( t_2 \) are \( e \)-bisimilar, but no \( e \)-bisimulation relates \( f(t_1) \) and \( f(t_2) \). In fact, both \( f(t_1) \) and \( f(t_2) \) have only one move, labelled \( a \) and taking to \( g(s_1, s_1) \) and \( g(s_2, s_2) \), respectively. Then, \( g(s_1, s_1) \) performs \( k(b, b) \) with probability \( \frac{1}{2} \), whereas \( g(s_2, s_2) \) performs \( k(b, b) \) with probability \( \frac{1}{2} + e + e^2 \), thus implying that \( g(s_1, s_1) \) and \( g(s_2, s_2) \), and, analogously, \( f(t_1) \) and \( f(t_2) \), can be related by an \( e \)-bisimulation only if \( e' \geq e + e^2 \). \( \blacktriangleright \)

Notice that in the example above the problem arises since two derivatives of the source variables \( x \), namely \( y_1 \) and \( y_2 \), appear in the conclusion of rule \( r_{ab}^f \). This is a form of duplication. A partial form of double testing may be allowed, meaning that premises \( \{ x \xrightarrow{\alpha_{k}} y_{k} \mid k \in K \} \) may be admitted, provided that only one of the variables \( y_{i,k} \) appears in the conclusion.

Next, we show why Definition 37 does not admit look ahead, i.e., active premises viewing two consecutive moves \( x \xrightarrow{\alpha_{i}} y_{j} \) and \( y_{j} \xrightarrow{\beta_{i}} z_{i} \) of any argument of \( f \). Intuitively, approximate bisimulation consider the maximal difference that can be evaluated on a single computation step, whereas look ahead would allow for accumulating the distances arriving in different computation steps, thus breaking non-expansiveness.

Example 28 (Look Ahead). Let \( f \) be the unary operator having the following rule \( r_{a_1,a_2} \), admitting look ahead, for all pairs of actions \( a_1, a_2 \in \mathcal{A} \):

\[
\frac{r_{a_1,a_2}^f}{x_1, x_2} \xrightarrow{\alpha_{1,2}, \beta_{1,2}} (x_1, x_2) \xrightarrow{\alpha_{2,1}} (x_1, x_2) \xrightarrow{\beta_{2,1}} (x_1, x_2) \xrightarrow{\gamma_{1,2}} (x_1, x_2)
\]

where \( k : A \times A \rightarrow A \) is a function injective in \((a, b), (a, c), (c, c)\). Let \( t_1 = a.(a +_j b +_j c) +_j c \) and \( t_2 = a.(a +_j b +_j c) +_j c \) be the processes in Fig. 6. Processes \( t_1 \) and \( t_2 \) are \( e \)-bisimilar, but there is no \( e \)-simulation relating \( f(t_1) \) and \( f(t_2) \). In fact, \( f(t_1) \) and \( f(t_2) \) perform action \( k(a, b) \) and reach nil with probability \( \frac{1}{2} + e + e^2 \) and \( \frac{1}{2} \), respectively. Hence, \( f(t_1) \) and \( f(t_2) \) can be related by an \( e' \)-bisimulation only if \( e' \geq e + e^2 \). \( \blacktriangleright \)

Now we recall the notion of afg-safe GTSS, i.e., a GTSS as in Definition 30 having only afg-safe rules and an additional constraint preventing behavioural distances generated by transitions with different actions labels to accumulate by giving rise to transitions with the same label.

Definition 38 (afg-safe GTSS, [110]). A GTSS as in Definition 30 is afg-safe if all rules are afg-safe, and

\[
\text{• for each action } a \in \mathcal{A}, \text{ let } R_a^{f,a} \text{ be the subset of the rules in } R_a^f \text{ having } a \text{ as action; the sum of the weights of the rules in any set } R_a^{f,a} \text{ is less than or equal to } 1.
\]

We can show that the clause in Definition 38 cannot be relaxed.

Example 29. Assume \( \mathcal{A} = \{a, b, c\} \) and let \( f \) be the unary operator having the following rules:

\[
\frac{r_a}{x} \xrightarrow{\alpha_p} y \quad \frac{r_b}{x} \xrightarrow{\beta_p} y \quad \frac{r_c}{x} \xrightarrow{\gamma_p} y
\]

The rules \( r_a \) and \( r_c \) violate Definition 38. Consider again the processes \( t_1 \) and \( t_2 \) in Fig. 5. They are \( e \)-bisimilar, but \( f(t_1) \) and \( f(t_2) \) cannot be related by any \( e \)-bisimulation, since they reach nil through \( b \) with probability \( \frac{1}{2} - 2e \) and \( \frac{1}{2} \), respectively, thus implying
that \( f(t_1) \) and \( f(t_2) \) can be related by a \( \epsilon' \)-bisimulation only if \( \epsilon' \geq 2\epsilon \). The point is that \( f \) maps both actions \( b \) and \( c \) of its process argument to action \( b \), so that the distance between \( f(t_1) \) and \( f(t_2) \) caused by \( b \) is the sum of the distance between \( t_1 \) and \( t_2 \) caused by \( b \) and that caused by \( c \). ▶

**Theorem 5 ([110]).** Let \( P \) be a \( \epsilon \)-safe RTSS. Then all generative \( \epsilon \)-bisimulations are non-expansive with respect to all operators defined by \( P \).

**Example 30 (Non expansive generative operators).** The constants \( \eta \) and \( a \), prefixing \( a.t \), probabilistic alternative composition \( t_1 +_p t_2 \), probabilistic interleaving \( t_1 \parallel_p t_2 \), ACP-like merge \( t_1 \parallel t_2 \) and restriction \( t \rightarrow A \) are defined in Table 3 and Table 4 by a set of rules respecting Definition 38 and, therefore, are non-expansive. On the other hand, in rules defining iterations \( t^\omega \), \( t^{\omega+1} \), \( t^* \), and replication \( \lambda t \), we observe replication of variables that violate the constraints in Definition 37; in the second rule for sequential composition \( t_1 : t_2 \) the derivative of the first argument does not appear in the target, thus violating the fourth clause in Definition 37; the rule for relabelling \( t[f] \) respects Definition 38 only if \( f \) is injective; the rules for priority \( h^\omega_i(t) \) have unquantified and negative premises violating the first clause in Definition 37. ▶

### 10.2. Non-expansiveness of approximate reactive bisimulations

We start with the notion of approximate reactive bisimulation safe (arb-safe) rule. Essentially, arb-safe rules are reactive rules as in Definition 12 that do not admit duplication of variables.

**Definition 39 (arb-safe rule, [110]).** A reactive rule as in Definition 12 is arb-safe if:

1. for each index \( i \in \{1, \ldots, n \} \setminus I \), at most one occurrence of source variable \( x_i \) appears in \( t \),
2. for each index \( i \in I \) at most one occurrence of derivative \( y_i \) of source variable \( x_i \) appears in \( t \), and, if no occurrence of \( y_i \) appears in \( t \), then at most one occurrence of \( x_i \) appears in \( t \).

Duplication of variables is forbidden for the same reason that was discussed in previous section for the generative case. The constraints in Definition 37 on unquantified and negative premises and that imposing that derivatives of source variables must appear in the target are not necessarily in the reactive case, since these constraints were imposed to deal with situations in which there is a probability in \((0,1)\) to perform a given action (cf. Example 24 and Example 26), which cannot happen in the reactive case. Double testing and look-ahead cannot be admitted, as in the generative case.

Now we recall the notion of arb-safe RTSS, i.e., a RTSS as in Definition 31 where all rules are arb-safe.

**Definition 40 (arb-safe RTSS, [110]).** A RTSS as in Definition 31 is arb-safe if all rules are arb-safe.

We note that the second item in Definition 38 is not needed in Definition 40 since it already appears in Definition 31 (see item 2b).

**Theorem 6 ([110]).** Let \( P \) be a arb-safe RTSS. Then all reactive \( \epsilon \)-bisimulations are non-expansive with respect to all operators defined by \( P \).

### 11. Non-expansiveness format for approximate bisimulations

In this section, we recall the format from [63] ensuring that bisimulations for the nondeterministic and probabilistic model are non-expansive. See also Table 9.

The approach in [63] can be summarised as follows. The first task is to assign an expansiveness bound to each operator. This is an upper bound on the distance between the composed processes, given the pairwise distances between their components. Specifically, the expansiveness bound on an \( n \)-ary operator \( f \) is defined as a mapping \( \text{exp}^f : \mathbb{R}^n \rightarrow \mathbb{R} \), such that \( \text{exp}^f(c_1, \ldots, c_n) = \epsilon \) if whenever \( t_i \sim t_i' \) for all \( i \in \{1, \ldots, n \} \), then \( f(t_1, \ldots, t_n) \sim \epsilon' f(t_1', \ldots, t_n') \) for some \( \epsilon' \leq \epsilon \). Expansiveness bounds are computed on the rules of a TSS. Then, the expansiveness bounds of the operators in a TSS are exploited to deduce a specification format ensuring non-expansiveness of approximate bisimulations. Notice that expansiveness bounds may be exploited also to deduce specification formats for other continuity properties.
The first observation in [63] is that the expansiveness of an operator \( f \) defined by a rule \( r \) depends on three factors: (i) the multiplicity (i.e. number of occurrences) of source variables and their derivatives in the target of \( r \), (ii) the expansive power of operators (i.e. how much does the operator multiply the distance of its arguments) that define a context in the rule target around the source variables or their derivatives, and (iii) the reactive behaviour discriminating power of the premises of \( r \).

**Example 31 (Factors of expansiveness).** Let \((\Sigma, A, R)\) be a PTSS with a signature \( \Sigma \) that contains constants \( t, s, nil \), unary operators \( f, f_2, \ldots \), binary operators \( g, g_2, \ldots \) and quaternary operators \( h, h_2, \ldots \), action set \( A = \{ a \} \), and axioms (i.e. rules without premises)

\[
\begin{align*}
    t & \rightarrow \delta_1 \\
    s & \rightarrow (1-\epsilon)\delta_1 + \epsilon\delta_{nil}
\end{align*}
\]

for some fixed \( \epsilon \in (0, 1) \). We have that \( t \sim_s s \) in the NPTS induced by \((\Sigma, A, R)\). Consider the rules:

\[
\begin{align*}
    x & \rightarrow \mu \\
    f(x) & \rightarrow g(\mu, \mu) \\
    x_1 & \rightarrow \mu_1 \\
    x_2 & \rightarrow \mu_2 \\
    g(x_1, x_2) & \rightarrow g(\mu_1, \mu_2)
\end{align*}
\]

These rules together with \( R \) define \( R_2 \). In the first rule the derivative \( \mu \) of source variable \( x \) appears twice in the rule target \( g(\mu, \mu) \). The induced NPTS of \((\Sigma, A, R_2)\) contains the following transitions:

![Diagram showing NPTS transitions](image)

Note that \( f(t) \sim_{q} f(s) \) for \( q = 1 - (1-\epsilon)^2 \). The power of 2 in the distance reflects directly the multiplicity of 2 of the derivative \( \mu \) in the rule target. The same effect can be observed for multiple occurrences of source variables in the rule target, e.g. consider for the \( f \)-defining rule \( g(\delta_1, \delta_1) \) instead of \( g(\mu, \mu) \) as target.

Furthermore, the expansive power of operators used in the rule target determine the expansiveness of the operator defined by that rule. A simple example is the axiom

\[
f_2(x) \rightarrow \delta_{f(s)}
\]

While the variable \( x \) occurs only once in the rule target, we still have \( f_2(t) \sim_{q} f_2(s) \) for \( q = 1 - (1-\epsilon)^2 \), because the operator \( f \) has an expansive power of 2 with respect to its single argument. This indicates that the expansive power of (arguments of) operators need to be defined recursively.

The multiplicity of source variables and their derivatives in the targets of the rules and the expansive power of operators applied on those variables multiply. Consider the rules:

\[
\begin{align*}
    x_1 & \rightarrow \mu_1 \\
    x_2 & \rightarrow \mu_2 \\
    g(x_1, x_2) & \rightarrow h(\mu_1, \mu_1, \mu_2, \mu_2) \\
    \{x_i \rightarrow \mu_i \mid i = 1, \ldots, 4\} & \rightarrow h(\mu_1, \mu_2, \mu_3, \mu_4)
\end{align*}
\]

These rules together with \( R_3 \) define \( R_3 \). Now \( f(t) \sim_{q'} f(s) \) for \( q' = 1 - (1-\epsilon)^3 \). As explained above for \( R_2 \), in the rule defining operator \( f \) the derivative \( \mu \) appears twice in the rule target. Additionally the operator \( g \) that is applied to \( \mu \) in the target of that rule has for both of its arguments an expansive power of two because in the \( g \)-defining rule in \( R_3 \) the derivatives \( \mu_1, \mu_2 \) of both arguments \( x_1, x_2 \) appear twice in the target.

The expansive power of an operator may be unbounded. Consider the operator \( f_3 \) defined by the rule:

\[
f_3(x) \rightarrow g(f_3(\mu), f_3(\mu))
\]

This rule together with \( R_2 \) define \( R_4 \). In the rule that defines \( f_3 \) the derivative \( \mu \) occurs twice in the target. Moreover, each occurrence of \( \mu \) is put in the context of that operator \( f_3 \) (recursive call). Additionally both occurrences of \( f_3(\mu) \) are put in the binary context \( g \), which enforces that the distances of the two copies of \( \mu \) multiply. Recursive multiplication of the distances leads to an approximate bisimulation distance of \( f_3(r) \sim_{f_3(s)} \). The expansive power of \( f_3 \) will in this case be denoted by \( \infty \).

On the other hand, an operator may absorb the approximate bisimulation distance. Consider the rules:

\[
\begin{align*}
    x & \rightarrow \mu \\
    f_4(x) & \rightarrow g_5(\mu, \mu) \\
    f_5(x) & \rightarrow g_6(\mu, \mu) \\
    g_4(x_1, x_2) & \rightarrow \delta_{nil}
\end{align*}
\]
These rules together with $R$ define $R_v$. Operator $g_3$ has no rule, operator $g_4$ allows for deriving an unconditional move to process $\text{nil}$. As a consequence, both $g_3$ and $g_4$ absorb the distance between their arguments, namely their expansive power is 0. Then, since in the rules for $f_1$ and $f_2$ the derivative $\mu$ of source variable $x$ appears in the context of an operator with the expansive power 0, also $f_3$ and $f_4$ absorb the distances of their arguments and their expansive power is 0. Indeed, $f_3(t) \sim_0 f_4(s)$ and $f_3(t) \sim_0 f_4(s)$.

If the reactive behaviour of the process associated to a source variable is tested by some premise, then the operator defined by this rule may discriminate states with different reactive behaviour. Consider rules:

$$
\begin{array}{c}
\frac{x \xrightarrow{\mu} \mu}{f_3(x) \xrightarrow{a} g_3(\mu, \mu)} \\
\frac{x_1 \xrightarrow{\mu_1} x_2 \xrightarrow{a} \mu_2}{g_3(x_1, x_2) \xrightarrow{a} \delta_d}
\end{array}
$$

These rules together with $R$ define $R_v$. Operator $g_3$ has the expansive power 0 whenever it is applied to pairs of process arguments $(s_1, s_2)$ and $(t_1, t_2)$ such that, for $i = 1, 2$, $s_i$ and $t_i$ have the same reactive behaviour, namely they have the same initial steps. Essentially, if both $s_i$ and $t_i$ perform $a_i$ or they are unable to make $a_i$ we have $g_3(s_1, s_2) \sim_0 g_3(t_1, t_2)$. Otherwise, we have $g_3(s_1, s_2) \not\sim_0 g_3(t_1, t_2)$. We will say that $g_3$ has reactive behaviour discriminating power, which has the same effect of the expansive power 1, whenever the process arguments have different reactive behaviour, and the same effect of the expansive power 0 otherwise. Then, since in the rule for $f_6$ we apply $g_5$ on top of 2 occurrences of the derivative $\mu$, the expansive power of $f_6$ is 2. Indeed, $f_6(t) \sim_q f_6(s)$ for $q = 1 - (1 - e)^2$. Note that also operators $f_4$ and $f_5$ defined above in $R_v$ have expansive power 0 and have reactive behaviour discriminating power.

We define by $\nu(f, i) \in \{0, 1\}$ the (reactive behaviour) discriminating power of argument $i$ of $f$.

**Definition 41 (Reactive behaviour discriminating power).** For a set of rules $R$, an operator $f$ and an argument $i$, we define $\nu(f, i) = 1$ if the source variable $x_i$ appears in a premise of some rule $r \in R_f$, i.e., if for some $r \in R_f$ there is a positive premise $x_i \xrightarrow{\mu}$ or a negative premise $x_i \xrightarrow{\mu'}$. Otherwise, $\nu(f, i) = 0$.

In [63] the expansive power of operators defined in a PTSS $(\Sigma, A, R)$ is computed by relying on a least fixed point of a monotone function. Let $\mathbb{N}^\omega$ denote $\mathbb{N} \cup \{\infty\}$, with the natural ordering extended by $n < \infty$ for each $n \in \mathbb{N}$, and the usual arithmetic extended for summation by $n + \infty = \infty + \omega = \infty = \infty + \infty = \infty$ for $n \geq 0$ and multiplication by $0 \cdot \infty = \infty \cdot 0 = 0$ and $\infty \cdot \infty = \infty \cdot \infty = \infty = \infty$ for $n \geq 1$.

We define a poset $S = (\Sigma, \sqsubseteq)$ with $S = S_\Sigma \times S_T$, $S_\Sigma = \Sigma \times \mathbb{N} \to \mathbb{N}^\omega$, $S_T = (\Sigma \times \mathcal{V}_p) \cup DT(\Sigma \times \mathcal{V}_p, V_d)$ into $(\mathcal{V}_d) \cup DT(\Sigma \times \mathcal{V}_p, V_d)$, equipped with the point-wise partial order $(m_2, m_T) \sqsubseteq (m_2', m_T')$ if and only if $m_2(f, i) \leq m_2'(f, i)$, for all $f \in \Sigma$ and $i \in \mathbb{N}$, and $m_T(t)(\zeta) \leq m_T'(t)(\zeta)$ for all $t \in \Sigma \times \mathcal{V}_p$ and $\zeta \in \mathcal{V}_d$. Elements of $S$ are pairs of maps $(m_2, m_T)$, where: (i) $m_2(f, i)$ denotes the expansive power of argument $i$ in operator $f$, i.e., how much the operator $f$ multiplies the approximate bisimulation distance of argument $i$; (ii) $m_T(t)(\zeta)$ defines the frequency of variable $\zeta \in \mathcal{V}_p \cup \mathcal{V}_d$ in the process or distribution term $t \in \Sigma \times \mathcal{V}_p \cup DT(\Sigma \times \mathcal{V}_p, V_d)$ weighted by the expansive power of the operators applied on top of $\zeta$.

$S$ forms a complete lattice with bottom element $\bot = (m_2^\bot, m_T^\bot)$ and top element $\top = (m_2^\top, m_T^\top)$, defined by constant maps $m_2^\bot(f, i) = 0$, $m_2^\top(f, i) = \infty$, $m_T^\top(t, \zeta) = 0$, $m_T^\bot(t, \zeta) = \infty$.

**Proposition 4 ([63]).** $S$ is a complete lattice.

We define now the *multiplicity functor* $M : S \to S$, which allows us to compute iteratively the expansive power of operators and the frequency of variables in terms.

**Definition 42 (Multiplicity functor).** Functor $M : S \to S$ is defined by $M(m_2, m_T) = (m_2', m_T')$, where:

$$
m_2'(f, i) = \sup_{r \in R_f} \left( m_2(\text{trgt}(r))(x_i) + \sum_{x_i \xrightarrow{a} \mu \in \text{prem}(r)} m_T(\text{trgt}(r))(\mu) \right);
$$

$$
m_T'(t)(\zeta) = \begin{cases} 
1 & \text{if } \zeta \in \mathcal{V}_p \text{ and } t = \zeta, \\
\sum_{i=1}^n (m_2(f, i) \cdot m_T(t_i)(\zeta)) & \text{if } t = f(t_1, \ldots, t_n), \\
0 & \text{otherwise;}
\end{cases}
$$

$$
m_T'(t)(\theta) = \begin{cases} 
1 & \text{if } \theta = \zeta, \\
\max_{\theta_i \in \mathcal{T}} (m_T'(\theta_i)(\zeta)) & \text{if } \theta = \sum_{i=1}^n \theta_i \cdot \theta, \\
\sum_{i=1}^n \max(m_2(f, i), \nu(f, i)) \cdot m_T'(\theta_i)(\zeta) & \text{if } \theta = f(\theta_1, \ldots, \theta_n), \\
0 & \text{otherwise.}
\end{cases}
$$
We can see that $M$ computes the expansive power of an operator $f$ by inspecting the rules for $f$ and taking into account: (i) the multiplicity of the variables in the target, (ii) the expansive power of the operators in the target, (iii) the reactive behaviour discriminating power of the operators in the target.

Functor $M$ is order-preserving. This ensures the existence and uniqueness of the least fixed point of $M$ by the Knaster-Tarski fixed point theorem.

**Proposition 5 ([63]).** $M$ is order-preserving.

We denote the least fixed point of $M$ by $(\omega_{<}, \omega_{\geq})$. We call $\omega_{<}(f, i)$ the expansive power of argument $i$ of operator $f$, and $\omega_{\geq}(t)(\xi)$ the weighted multiplicity of variable $\xi$ in term $i$. The expansive power of $f$ allows us to derive an upper bound on the approximate bisimulation distance between terms $f(t_{1}, \ldots, t_{n})$ and $f(t'_{1}, \ldots, t'_{n})$ expressed in relation to the approximate bisimulation distances $e_{i}$ between their arguments.

**Definition 43 (Expansiveness bound).** The expansiveness bound $\exp^{f}$ of operator $f \in \Sigma$ with respect to the approximate bisimulation distances $e_{i}$ of its arguments $i = 1, \ldots, n$ is defined by

$$\exp^{f}(e_{1}, \ldots, e_{n}) = 1 - \prod_{i=1}^{n}(1 - e_{i})^{\omega_{<}(f, i)}$$

The pattern $1 - \prod_{i=1}^{n}(1 - e_{i})^{\omega_{<}(f, i)}$ can be explained as follows. In the derivatives of $f(t_{1}, \ldots, t_{n})$, $\omega_{<}(f, i)$ copies of the derivatives of process argument $t_{i}$ are created. Therefore, $\omega_{<}(f, i)$ instances of $t_{i}$ and $t'_{i}$ contribute with their distance $e_{i}$ to the distance between $f(t_{1}, \ldots, t_{n})$ and $f(t'_{1}, \ldots, t'_{n})$. These $\omega_{<}(f, i)$ constructs are not simply summed: in the pattern $1 - (1 - e_{i})^{\omega_{<}(f, i)}$ the distance $e_{i}$ between one occurrence of $t_{i}$ and one occurrence of $t'_{i}$ is weighted by the probability that all other occurrences of $t_{j}$ and $t'_{j}$, for all $j \in [1, n]$, behave in the same way.

**Example 32 (continued).** For the PSOS TSS ($\Sigma, A, R_{3}$) we have $\omega_{<}(f, 1) = 2$ because $\omega_{<}(g, 1) = \omega_{<}(g, 2) = 1$. We continue with the TSS ($\Sigma, A, R_{3}$). For operator $h$ we have $\omega_{<}(h, i) = 1$ for $i = 1, \ldots, 4$, for $g$ we have $\omega_{<}(g, 1) = \omega_{<}(g, 2) = 2$ and thus for $f$ we get $\omega_{<}(f, 1) = 4$. For the TSS ($\Sigma, A, R_{3}$) the recursive definition of $f$ applied to the two occurrences of the derivative $\mu$ in the rule target gives $\omega_{<}(f, 1) = \infty$. For TSS ($\Sigma, A, R_{3}$) we get $\omega_{<}(f_{1}, 1) = \omega_{<}(f_{2}, 1) = \omega_{<}(g_{4}, 1) = \omega_{<}(g_{4}, 2) = 0$. Finally, in the TSS ($\Sigma, A, R_{3}$) the (reactive behaviour) discriminating power $\gamma(g_{3}, 1) = \gamma(g_{3}, 2) = 1$ of operator $g_{3}$ leads to $\omega_{<}(f_{6}, 1) = 2$. ▶

The following result states that the expansiveness bound of operators can be exploited to obtain an upper bound on the approximate bisimulation distance between processes. The result was given in [63] for simple ntrfθ/ntfxθ-TSSs [40,42,63] recalled in Definition 33.

**Theorem 7 ([63]).** Let ($\Sigma, A, R$) be a PGSOS-PTSS. Then for each operator $f \in \Sigma$ we have

$$f(t_{1}, \ldots, t_{n}) \sim_{c} f(t'_{1}, \ldots, t'_{n}) \quad \text{if} \quad t_{i} \sim_{c} t'_{i} \quad \text{for} \quad i = 1, \ldots, \tau(f) \quad \text{and} \quad c = \exp^{f}(e_{1}, \ldots, e_{n}).$$

Notice that Theorem 7 implies the congruence of [40,42] for probabilistic bisimilarity, by considering $e_{i} = 0$ for all $i = 1, \ldots, n$ and exploiting that $\sim_{c}$ is in fact the strict probabilistic bisimilarity.

Clearly, Theorem 7 allows one to infer the uniform continuity properties of operators from the rules, but computing $\exp^{f}$ requires a recursive reasoning over those rules. In [63] a rule format for non-expansiveness was proposed, which exploits Theorem 7 and whose constraints on the rules are easy to verify, since it suffices to count the occurrences of source variables and derivatives in the rule target without any need for recursive reasoning over other rules.

Let $\text{MVar} : (\Sigma, V_{c}) \cup \text{DT} \Sigma, V_{c} (V_{c}) \rightarrow (V_{c} \cup V_{t} + \Sigma)$ denote for $\text{MVar}(\theta)(\xi)$ how often the variable $\xi$ occurs in $\theta$. Formally, for process terms we define $\text{MVar}(x)(\xi) = 1$ if $x = \xi$, $\text{MVar}(x)(\xi) = 0$ if $x \neq \xi$, and $\text{MVar}(f(t_{1}, \ldots, t_{n}))(\xi) = \sum_{i=1}^{n} \text{MVar}(t_{i})(\xi)$. Then for distribution terms we define

- $\text{MVar}(\mu)(\mu) = 1$ and $\text{MVar}(\mu)(\xi) = 0$ if $\mu \neq \xi$,
- $\text{MVar}(\delta_{\xi})(\xi) = \text{MVar}(\delta_{\xi})(\xi)$,
- $\text{MVar}(\sum_{i=1}^{n} p_{i} \theta_{i})(\xi) = \max_{i=1}^{n} \text{MVar}(\theta_{i})(\xi)$, and
- $\text{MVar}(f(\theta_{1}, \ldots, \theta_{n}))(\xi) = \sum_{i=1}^{n} \text{MVar}(\theta_{i})(\xi)$.

From the expansiveness bound $\exp^{f}$ of operator $f \in \Sigma$ it follows that $f$ is non-expansive if $\omega_{<}(f, i) \leq 1$ for all $i = 1, \ldots, n$. This yields the following rule format.

**Definition 44 ($e$-PGSOS rule format).** A PGSOS-rule is an $e$-PGSOS rule if for each $x_{i} \in \text{src}(r)$ we have

$$\text{MVar}(\text{trgt}(r))(x_{i}) + \sum_{x_{i} \in \text{src}(r)} \text{MVar}(\text{trgt}(r))(\mu) \leq 1$$

34
A PTSS $P = (\Sigma, A, R)$ is in $\epsilon$-PGSOS format if all rules in $R$ are in the $\epsilon$-PGSOS format.

**Theorem 8** ([I63]). Let $(\Sigma, A, R)$ be a $\epsilon$-PGSOS-PTSS. Then all operators $f \in \Sigma$ are non-expansive.

**Example 33.** By removing from Table 2 all operators allowing for iteration we get an $\epsilon$-PGSOS-PTSS. ▶

**Remark 11.** Let us consider the sequential composition operator $\cdot$, whose rules are given in Table 2. This operator describes separate moves of either the first component or the second one. Hence, the approximate bisimulation distance (being the maximum distance arising along the computation path) is determined by only one of them. In other words, $\cdot$ is non-extensive, i.e., its expansiveness is bounded by $1 - \min\{1 - \varepsilon_1, 1 - \varepsilon_2\} = \max\{\varepsilon_1, \varepsilon_2\}$. However, according to Definitions 42 and 43, sequential composition has expansive power $\omega_2(\cdot)(1) = \omega_2(\cdot)(2) = 1$, giving an expansiveness bound $\exp\{\varepsilon_1, \varepsilon_2\} = 1 - (1 - \varepsilon_1)(1 - \varepsilon_2)$, which means that the distances of the arguments are summed up. We plan, as future work, to refine the non-expansive format discussed in this section, in order to provide a proper characterisation of non-extensive operators.

12. Formats highlights

In this part we focused on non-expansiveness formats for the approximated variants of generative, reactive, and probabilistic bisimilarity. In each case, the format is obtained by adding additional constraints to the formats ensuring the congruence property of the respective bisimulation relation. This is due to the fact that each of the three $\epsilon$-bisimulation relations coincides with the non-approximated variant when $\epsilon = 0$.

The format for generative $\epsilon$-bisimulation, the agb-safe GTSS format (Definition 38), is based on the notion of agb-safe rules, i.e., generative rules that: (i) admit only a limited form of unquantified and negative premises, (ii) do not admit duplication of variables, excluding thus also double testing and look-ahead, and (iii) force the derivatives of source variable to appear in the target. Moreover, the format includes an additional constraint on the partition of the rules defining an operator that prevents behavioural distances generated by transitions with different actions labels to accumulate by giving rise to transitions with the same label.

The arb-safe RTSS format (Definition 40) is obtained in a similar fashion, although less restrictive requirements are needed. Specifically, the arb-safe rules on which it is built, only forbid the duplication of variables, since the additional constraints in the agb-safe rules were necessary to manage the case in which the probability to perform a given action is in $(0, 1)$, which cannot happen in the reactive case. Moreover, the constraints on the partitions of rules for the pairs operator, action given in the RTSS format directly prevent the accumulation of behavioural distances in the derivation of transitions.

The case of the $\epsilon$-PGSOS format (Definition 44) is different, since the patterns of the PGSOS rules are inspected to quantify the expansiveness bound of operators, namely an upper bound on how much the original distance between processes is expanded when they are composed by means of a specific operator. These bounds are obtained by inspecting how many times variables are duplicated in the rules for that operator, and the expansive bounds of the operators that appear in the targets of those rules.

13. Continuity formats for bisimilarity metrics over NPTs

Formats for non-extensiveness, non expansiveness and Lipschitz continuity for bisimilarity metrics over NPTs can be found in [34,64,66], and are also reported in Table 10. However, the approach used in [64,66] for the definition of the formats is not fully syntactic as, besides inspecting the pattern of the rules, it requires to compute how many copies of a process argument can be spawned by each operator. Hence, to remain close to the fully syntactic approach of the previous sections, we will discuss the formats proposed in [34].

Our first observation is that, intuitively, uniform continuity requires that the number of copies of process arguments that can be created along the computation should be bounded.

**Example 34 (Unlimited copies of processes).** Assume that $\mathcal{A}$ contains the set of actions $\{a_k | k \in \mathbb{N}\}$. Let us define the operators $f, g, h$ by means of the following rules:
Operators $f$ and $g$ can create an arbitrary number of copies of their process argument. We can show that $f$ and $g$ are not uniformly continuous. To this purpose, let us consider processes $s_k = a_k.a_k.nil$ and $t_k = a_k.((1 - e) a_k.nil @ e.nil)$, with $k \in \mathbb{N}$. Clearly, $b^k(s_k, t_k) = kc$ for all $k \in \mathbb{N}$.

Consider first operator $f$. We have $f(s_k) = \pi_{t_k}$ and $f(t_k) = \pi_{s_k}$, with $\pi_{t_k}$ and $\pi_{s_k}$ distributions such that $\pi_{t_k}(a_k.nil \|_{A} \ldots \|_{A} a_k.nil) = 1$, $\pi_{s_k}(a_k.nil \|_{A} \ldots \|_{A} a_k.nil) = (1 - e)^k$ and $\pi_{t_k}$ giving the remaining probability $1 - (1 - e)^k$ to processes that cannot move. We get $b^k(f(s_k), f(t_k)) = \lambda t_k \epsilon (1 - (1 - e)^k)$. Hence, $\sup_{k \in \mathbb{N}} b^k(f(s_k), f(t_k)) = \sup_{k \in \mathbb{N}} \lambda (1 - (1 - e)^k) = \lambda$, thus giving that the distance between $f$-composed processes is not bounded by $m(c) = k$ if $e > 0$ and $m(0) = 0$, which is not a modulus of continuity since it is not continuous at 0. Hence, $f$ is not uniformly continuous accordingly to Definition 29.

Consider now operator $g$. We have $g(s_k) = \delta_{t_k...t_k}(s_k)$ (with $k$ nested occurrences of $h$) and $g(t_k) = \delta_{s_k...s_k}(t_k)$... Then, $h(...h(s_k)...)(...h(t_k)...)$ gives $\pi_{t_k}'$ and $h(...h(s_k)...)(...h(t_k)...)$ gives $\pi_{s_k}'$, with $\pi_{t_k}'(a_k.nil \|_{A} \ldots \|_{A} a_k.nil) = (1 - e)^k$ and $\pi_{s_k}'$ giving probability $1 - (1 - e)^k$ to processes that cannot move. We get $b^k(g(s_k), g(t_k)) = \lambda^2 (1 - (1 - e)^k)$ and, then, $\sup_{k \in \mathbb{N}} b^k(g(s_k), g(t_k)) = \sup_{k \in \mathbb{N}} \lambda^2 (1 - (1 - e)^k) = \lambda^2$. As for operator $f$, we conclude that no modulus of continuity for $g$ can be given and, therefore, $g$ is not uniformly continuous.

Our second observation is that, intuitively, Lipschitz continuity requires that the number of copies of process arguments that can be created along the computation should be bounded by a constant.

**Example 35 (Copy operator).** Consider the copy operator $[27]$ defined by the following rule:

\[
\begin{array}{l}
\xrightarrow{a} x &\text{if } a \notin \{l, r\} \\
\xrightarrow{l} x &\xrightarrow{r} y \\
\xrightarrow{a} x &\xrightarrow{v}
\end{array}
\]

\[\text{cp}(x) \xrightarrow{a} \mu \quad \text{cp}(x) \xrightarrow{v} \mu
\]

The second rule allows us to create an exponential number of copies of process argument. We can show that this breaks Lipschitz continuity for any candidate Lipschitz factor $L$.

Let $e \in (0, 1)$. Consider processes $s_1 = l.a.nil + r.a.nil$ and $t_1 = l.((1 - e)a.nil @ e.nil) + r.((1 - e)a.nil @ e.nil)$, for $a \neq l, r$ Then, for $k > 1$, let $s_k = l.s_k + r.s_k$ and $t_k = l.t_k$. Clearly, $b^k(s_k, t_k) = \lambda^k e$. By performing a sequence of $k$-labelled transitions, process $cp(s_k)$ reaches a distribution that assigns probability 1 to the parallel composition of $2^k$ copies of process $a.nil$. By performing the same sequence of transitions, $cp(t_k)$ reaches a distribution assigning probability $(1 - e)^k$ to the same process, and the remaining probability $1 - (1 - e)^k$ to processes that cannot execute any action (being the parallel composition of $2^k-1$ processes among which there is at least one process). Therefore we get $b^k(cp(s_k), cp(t_k)) = \lambda^k (1 - (1 - e)^k)$. This implies that if we take any $k$ with $2^k > L$, and $e$ is such that $e \in (0, (2^k - L)/(2^{k-1} - 2^k - 1))$, then we get $b^k(cp(s_k), cp(t_k))/b^k(s_k, t_k) = (1 - (1 - e)^k)/e > L$.

Since processes are at bisimulation distance zero if and only if they are bisimilar (Proposition 1), a format guaranteeing uniform continuity for bisimilarity metric must also guarantee the congruence property for bisimilarity. In Section 8 we argued that the congruence property for bisimilarity is guaranteed by the $n_{t\mu}f/nt\mu x\theta$ format of $[40,42]$. However, the following example shows that already the simple version of the $n_{t\mu}f/nt\mu x\theta$ format (Definition 33) breaks Lipschitz continuity, thus implying that when dealing with metrics we cannot go beyond PGOS. Intuitively, by using operators in the source of the premises of the rules, if those operators are not non-expansive, then we can create an exponential number of copies of the derivatives of source variables, which breaks Lipschitz continuity as already observed above.

**Example 36 (ntu\theta/ntu x\theta rules).** Let us extend Table 2 with a new operator $f$ defined as follows:

\[
\begin{array}{l}
x \parallel_{A} x \xrightarrow{a} \mu \\
f(x) \xrightarrow{a} f(\mu)
\end{array}
\]

The rule for $f$ is simply-ntu\theta but falls out of the PGOS format since a term different from a single variable occurs as left hand side of the premise. We can show that $f$ cannot be $L$-Lipschitz continuous for any fixed constant $L$. Consider processes $s_1 = a.a.nil$ and $t_1 = a.((1 - e)a.nil @ e.nil)$. Then, for all $k > 1$ define $s_k = a.s_k$ and $t_k = a.t_k$. Clearly, for all $k \geq 1$ we have $b^k(s_k, t_k) = \lambda^k e$. We now aim at evaluating $b^k(f(s_k), f(t_k))$, for all $k \geq 1$. Since the target of the rule for $f$ is of the form $f(\mu)$, where $\mu$ is derived from the synchronous parallel composition of the variable with itself, we can infer that $f(s_k)$ via a sequence of a $k$-labelled transitions reaches a distribution that assigns probability 1 to the process obtained by applying $f$ to the parallel composition of $2^k$ copies of process $a.nil$. Similarly, after $k$-labelled transitions $f(t_k)$ reaches a distribution assigning probability $(1 - e)^k$ to the same process and the remaining probability $1 - (1 - e)^k$ to processes that cannot execute any action (being obtained by applying $f$ to the parallel composition of $2^k-1$ processes among which there is at least one process nil). Therefore we get $b^k(f(s_k), f(t_k)) = \lambda^k (1 - (1 - e)^k)$. Already in Example 35 we argued that $b^k(s_k, t_k) = \lambda^k e$ and $b^k(f(s_k), f(t_k)) = \lambda^k (1 - (1 - e)^k)$. break $L$-Lipschitz continuity for suitable choices of $k$ and $e$. ◀
Next, we need to understand how we can obtain the upper-bounds on the distance, corresponding to the moduli of continuity from Definition 29, from the syntactic inspection of the rules. To this end, consider the fix-point characterisation of the bisimilarity metric (Proposition 3). By monotonicity of the suprema and infima used in the definition of the bisimulation metric functional (Definition 26), we can transfer any upper-bound on the Kantorovich lifting to an upper-bound on $b^t$. In particular, we can exploit the following general properties of the Kantorovich distance, that hold for any 1-bound pseudometric $d$ [66]:

$$K(d)(\delta_\omega, \delta_v) = d(t, u), \text{ for all } t, u \in T(\Sigma), \text{ and } K(d)(\sum_{j \in J} p_j \epsilon_j, \sum_{j \in J} p_j \epsilon_j') \leq \sum_{j \in J} p_j \cdot K(d)(\epsilon_j, \epsilon_j').$$

(2)

Since the distributions over which the Kantorovich distance is evaluated are obtained as closed instances of the targets of the PGSOS rules, the uniform continuity formats will be characterised by various constraints on the form of those distribution terms.

13.1. Non-extensiveness format

We start with PGSOS rules for non-extensiveness, which admit only convex combinations of variables and constants in the rule target. Remind that $\Sigma^n$ denotes the subset of the n-ary operators in $\Sigma$, and let $\delta_\nu$ denote the set of distribution terms $\delta_\nu = \{\delta_\nu \mid x \in \nu \}$.

**Definition 45 (Non-extensive TSS, [34]).** We say that a PGSOS TSS $P = (\Sigma, A, R)$ (Definitions 10 and 32) is non-extensive if the target of each rule is of the form $\theta = \sum_{j \in J} p_j \theta_j$, with $\theta_j \in \nu \cup \delta_\nu \cup \Sigma^0$ for all $j \in J$. All operators in $\Sigma$ are called $P$-non-extensive.

Essentially, in each distribution term $\theta_j$ we have at most one occurrence of one source variable $x_i$ or of one of its derivatives. Moreover, that variable does not appear in the context of any operator. Intuitively, this implies that $\theta_j$ contributes to the distance $b^t(f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n))$ by at most $b^t(t_i, t'_i)$. By Equation (2) this gives $b^t(f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n)) \leq \sum_{j \in J} p_j \max_{\omega=1}^n b^t(t_i, t'_i)$.

**Theorem 9 (Non-extensiveness, [34,66]).** Let $\lambda \in (0,1)$ and $P = (\Sigma, A, R)$ be a non-extensive TSS. Then, all operators in $\Sigma$ are non-extensive with respect to $b^t$.

Notice that a PTSS with only the operators $a.t, t + t', t + t'$ and $a.\bigoplus_{i=1}^n [a_i]t_i$ is non-extensive. In [62] it is shown that all the remaining binary operators in Table 2, which are out of Definition 45, are not non-extensive, thus witnessing that the constraints in Definition 45 are not too demanding.

**Example 37.** All rules for operator $\|_A$ are out of Definition 45, since $\|_A$ appears in the target. Let $t_1 = a.((0.9)b.nil \oplus (0.1)nil)$ and $t_2 = a.b.nil$ be the processes in Fig. 7. We have $t_1 \xrightarrow{a} 0.9 b_{nil} + 0.1 b_{nil}$ and $t_2 \xrightarrow{a} b_{nil}$. We get $b^t(t_1, t_2) = \lambda \cdot K(b^t(0.9 b_{nil} + 0.1 b_{nil}, b_{nil})) = \lambda \cdot 0.1$. Note that 0.1 is the probability that $t_1$ cannot perform $b$ at second step. Consider $A = \{a, b\}$. We have $t_1 \|_A t_1 \xrightarrow{b} 0.8 1 b_{nil} \oplus 0.09 b_{nil} \oplus 0.09 b_{nil} \oplus 0.09 b_{nil} \oplus 0.01 a_{nil} \oplus 0.1 a_{nil}$ and $t_2 \|_A t_2 \xrightarrow{b} b_{nil} \oplus b_{nil} \oplus b_{nil} \oplus b_{nil} \oplus b_{nil}$ (Fig. 7). We get $b^t(t_1 \|_A t_1, t_2 \|_A t_2) = \lambda \cdot 0.19$, with 0.19 the probability that $t_1 \|_A t_1$ can not perform $b$ at second step.

13.2. Non-expansiveness format

We obtain a format for non-expansiveness by relaxing Definition 45 as follows. We can allow a rule target to be a convex combination of distribution terms in which also non-constant operators occur and are applied to source variables and their derivatives, provided that, for each source variable, at most one occurrence of that variable, or its derivatives, occurs.

**Definition 46 (Non-expansive TSS, [34]).** Let $P_1 = (\Sigma_1, A, R_1)$ be a non-extensive TSS. A PGSOS TSS $P_2 = (\Sigma_2, A, R_2)$ with $P_1 \subseteq P_2$ is non-expansive if all rules $r \in R_2 \setminus R_1$ are PGSOS rules in which:

1. the target $\theta$ is of the form $\theta = \sum_{j \in J} p_j \theta_j$;
2. for all \( j \in J \) and \( i \in I \), at most one among the variables \( \{ x_i \} \cup \{ \mu_{i,k} \mid k \in K_i \} \) occurs in \( \theta_j \) and there must be at most one such occurrence;
3. for all \( j \in J \) and \( i \in \{1, \ldots, n\} \setminus I \), the variable \( x_i \) occurs at most once in \( \theta_j \).

All operators defined by rules in \( R_2 \setminus R_1 \) are called \( P_2 \)-non-expansive.

Since in each \( \theta_j \) we can have at most one occurrence of a source variable \( x_i \) or of one of its derivatives, and that variable may appear in the context of operators that are non-expansive and cannot amplify the distance between their process arguments, we infer that \( \theta_j \) contributes to \( b^i(f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n)) \) by, at most, \( \sum_{i=1}^n b^i(t_i, t'_i) \). Consequently, by Equation (2) we get that \( b^i(f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n)) \leq \sum_{j \in J} p_j \sum_{i=1}^n b^i(t_i, t'_i) = \sum_{i=1}^n b^i(t_i, t'_i) \).

**Theorem 10** (Non-expansiveness, [34,66]). Let \( \lambda \in (0,1) \) and \((\Sigma, A, R)\) be a non-expansive TSS. Then the operators in \( \Sigma \) are non-expansive with respect to \( b^i \).

Notice that all operators for parallel composition and operators \( \tau, t', f(f), \land A \) and \( \theta^i(t) \) are non-expansive. Since the last three operators are unary, they are non-expansive. Conversely, in [62] it is showed that all operators in Table 2 allowing for iteration, which are not admitted by Definition 46, are not non-expansive, which confirms that the constraints in Definition 46 are not too demanding.

**Example 38.** The rule for \( 'A \), given in Table 2, violates Definition 46 since both the source variable \( x \) and its derivative \( \mu \) occur in the same distribution term in the target. Consider \( A = \{ b \} \), \( t'_1 = t_1 + b.nil \) and \( t'_2 = t_2 + b.nil \), with \( t_1 \) and \( t_2 \) as in Fig. 7. We have \( b^i(t'_1, t'_2) = b^i(t_1, t_2) = \lambda \cdot 0.1 \), since \( t'_j \) has the same \( a \)-move of \( t_i \) and \( t'_j \) has the same \( a \)-move of \( t_i \). For \( i = 1, 2 \), consider the \( a \)-transitions performed by \( !A t'_1 \) and \( !A t'_2 \). We have \( !A t'_1 \xrightarrow{a} x_1 = 0.9\delta_{b.nil}|t'_1|x'_1 + 0.1\delta_{b.nil}|t'_1|x'_2 \) and \( !A t'_2 \xrightarrow{a} x_2 = 0.9\delta_{b.nil}|t'_2|x'_1 + 0.1\delta_{b.nil}|t'_2|x'_2 \). Then, the processes in the support of \( x_1 \) and \( x_2 \) have the \( a \)-steps \( b.nil|t'_1|x'_1 \xrightarrow{a} x'_1 = 0.9\delta_{b.nil}|b.nil|t'_1|x'_1 + 0.1\delta_{b.nil}|b.nil|t'_1|x'_2 \), where only the first process in the support of \( x'_1 \) can perform \( b \) and \( \delta_{b.nil}|t'_1|x'_1 \xrightarrow{a} x'_2 = 0.9\delta_{b.nil}|b.nil|t'_1|x'_1 + 0.1\delta_{b.nil}|b.nil|t'_1|x'_2 \), where none of the two processes in the support of \( x'_2 \) can perform \( b \). Summarising, after two \( a \)-steps, \( !A t'_1 \) and \( !A t'_2 \) have probability to perform \( b \) which is, respectively, 0.81 and 1. We infer that \( b^i(t'_1, t'_2) \geq \lambda^2 \cdot (1 - 0.81) = \lambda^2 \cdot 0.19 \). Since we had \( b^i(t'_1, t'_2) = \lambda \cdot 0.1 \), we can conclude that \( !A \) is non-expansive for any \( \lambda \) such that \( 0.19^2 > 0.1 \).

### 13.3. Lipschitz-continuity format

A format for Lipschitz continuity can be obtained by relaxing Definition 46 as follows. Firstly, for each operator \( f \) we fix two natural numbers \( n_f, m_f \in \mathbb{N} \). Then, we allow the targets of the rules for \( f \) to be convex combinations of distribution terms in which: (i) the number of occurrences of source variables and their derivatives is bounded by \( n_f \); (ii) the number of nested operators in which a source variable can occur is bounded by \( m_f \); (iii) derivatives of source variables can only occur in the scope of non-expansive operators.

**Definition 47** (Lipschitz TSS, [34]). Let \( P_1 = (\Sigma_1, A, R_1) \) be a non-expansive TSS. A PGSOS TSS \( P_2 \) = \( (\Sigma_2, A, R_2) \) with \( P_1 \subset P_2 \) is Lipschitz if for all operators \( f \in \Sigma_2 \setminus \Sigma_1 \) there are naturals \( n_f, m_f \in \mathbb{N} \) and all rules \( r \in R_2 \setminus R_1 \) for \( f \) are PGSOS rules in which:

1. the target \( \theta \) is of the form \( \theta = \sum_{i \in J} p_i \theta_i \),
2. for all \( j \in J \) and \( i \in \{1, \ldots, n\} \), the source variable \( x_i \) occurs at most \( n_f \) times in \( \theta_j \) and such occurrences appear in the scope of at most \( m_f \) operators,
3. for all \( j \in J \) and \( i \in I \), there are at most \( n_f \) occurrences of variables in \( \{ \mu_{i,k} \mid k \in K_i \} \) in \( \theta_j \), which appear only in the scope of non-expansive operators in \( \Sigma_1 \).

Then, all operators defined by rules in \( R_2 \setminus R_1 \) are called \( P_2 \)-Lipschitz.

Notice that the rules in Table 2 for operators \( r^i, t^r \) and \( \!A t^r \) that were ruled out by Definition 46, respect Definition 47.

Definition 47 allows for deriving a relation between the Lipschitz factors of the operators. Given a source variable \( x_i \) and a distribution term \( \theta_j \); (a) clause 3 limits to \( n_f \) the number of occurrences of the derivatives of \( x_i \) that may occur in \( \theta_j \), and requires them to be in the scope of non-expansive operators, which we know, cannot amplify the distance between their process arguments. This ensures that the derivatives of \( x_i \) in \( \theta_j \) contribute to \( b^i(f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n)) \) by, at most, \( p_j \cdot n_f \cdot b^i(t_i, t'_i) \); (b) clause 2 limits to \( n_f \) the number of occurrences of \( x_i \) that may occur in \( \theta_j \) and guarantees that at most \( m_f \) operators are applied on top of them, thus ensuring that if we name by \( L^i_{\theta_j} \) the maximum of the Lipschitz factors of these \( m_f \) operators, then we infer that \( x_i \) and \( \theta_j \) contribute to distance \( b^i(f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n)) \) by, at most, \( p_j \cdot \lambda \cdot n_f \cdot L^i_{\theta_j} \cdot b^i(t_i, t'_i) \). Notice that factor \( \lambda \) arises since \( x_i \) is delayed by one computation step. Summarising, from (a) and (b) we infer that \( x_i \) and \( \theta_j \) contributes to \( b^i(f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n)) \) by, at
most, \( p_j \cdot n_j \cdot (1 + \lambda \cdot L_\theta^{\text{max}}) \cdot \sum_{i=1}^n b^i(t_j, t_j') \). Hence, \( b^i(f(t_1, \ldots, t_n), f(t_1', \ldots, t_n')) \leq \sum_{j \in \Theta} p_j \cdot n_j \cdot (1 + \lambda L_\theta^{\text{max}}) \cdot \sum_{i=1}^n b^i(t_j, t_j') \). This implies that 

\( L_f \) should be at least \( \sum_{j \in \Theta} p_j \cdot n_j (1 + \lambda L_\theta^{\text{max}}) \), where we remind that \( L_\theta \) is an expression in which Lipschitz factors may appear.

**Definition 48** (Lipschitz factor, [34]). Assume \( \lambda \in (0, 1) \), a non-expansive TSS \( P_1 = (\Sigma_1, A, R_1) \) and a TSS \( P_2 = (\Sigma_2, A, R_2) \) with \( P_1 \subseteq P_2 \).

For each operator \( f \in \Sigma \subseteq \Sigma_1 \) let \( R_j \subseteq R_2 \) be the set of rules defining its behaviour. Then, the set of Lipschitz factors of operators in \( \Sigma \) is defined as the set of minimum reals \( L_f \geq 1 \) satisfying the system of inequalities of the form

\[
L_f \geq \max_{r \in \Theta_f} \max_{i \in \{1, \ldots, n\}} \left\{ \ell_\theta(x_i) + \sum_{(j, \mu) \in \Theta_f} \ell_\theta(\mu_j) \right\}
\]

for each \( f \in \Sigma_1 \setminus \Sigma_1 \), where \( \theta_j \) denotes the target of \( r \) and for any distribution term \( \theta \in \Delta(T(\Sigma, \nu, \nu)) \), the factor \( \ell_\theta \) is defined inductively as follows:

\[
\ell_\theta(x) = \begin{cases} 
0 & \text{if } x \text{ does not occur in } \theta \\
\lambda & \text{if } \theta = \delta_x \\
L_f \sum_{i=1}^n \ell_\theta(x_i) & \text{if } \theta = f(\theta_1, \ldots, \theta_n) \text{ and } f \in \Sigma_1 \\
\sum_{i \in \Theta} p_i \ell_\theta(x_i) & \text{if } \theta = f(\theta_1, \ldots, \theta_n) \text{ and } f \in \Sigma_1 \\
0 & \text{if } \mu \text{ does not occur in } \theta \\
1 & \text{if } \theta = \mu
\end{cases}
\]

\[
\ell_\theta(\mu) = \begin{cases} 
\max_{i=1}^n \ell_\theta(\mu_i) & \text{if } \theta = f(\theta_1, \ldots, \theta_n) \text{ and } f \in \Sigma_1 \\
L_f \sum_{i=1}^n \ell_\theta(\mu_i) & \text{if } \theta = f(\theta_1, \ldots, \theta_n) \text{ and } f \in \Sigma_1 \\
\sum_{i \in \Theta} p_i \ell_\theta(\mu_i) & \text{if } \theta = f(\theta_1, \ldots, \theta_n) \text{ and } f \in \Sigma_1 \\
0 & \text{if } \mu \text{ does not occur in } \theta \\
1 & \text{if } \theta = \mu
\end{cases}
\]

**Proposition 6** ([34]). Let \( \lambda \in (0, 1) \). Assume a non-expansive TSS \( P_1 = (\Sigma_1, A, R_1) \) and a Lipschitz extension \( P_2 = (\Sigma_2, A, R_2) \). Then, for each operator \( f \in \Sigma_2 \), the Lipschitz factor \( L_f < +\infty \) is well-defined.

Notice that for \( op \) any operator in \( r^*, r^o \) and \( \lambda \), we get \( L_{op} \geq \lambda L_{op} + 1 \), which gives \( L_{op} \geq \frac{1}{1 - \lambda} \) in case \( \lambda < 1 \).

**Theorem 11** (Lipschitz continuity, [34]). Let \( (\Sigma, A, R) \) be a Lipschitz-TSS and let \( \lambda \in (0, 1) \). Then the operators in \( \Sigma \) are Lipschitz continuous with respect to \( b^i \). In particular, for a Lipschitz operator \( f \), we have that \( f \) is \( L_f \)-Lipschitz with respect to \( b^i \).

Summarising, operators \( r^*, r^o \) and \( \lambda \) in Table 2 are Lipschitz continuous if \( \lambda < 1 \), \( r \) and all operators for parallel composition are non-expansive, \( a, t, t^t, t^t \) and \( a \) with \( |a| \leq 1 \) are non-expansive.

We conclude by showing how the operators in the examples of this section that were shown to be non-Lipschitz continuous do not respect Definition 47. Operator \( f \) in Example 34 violates clause 3 since no \( n_f \) can be given. Operator \( g \) violates clause 2 since no \( m_g \) can be given. In Example 35, the second rule for operator \( cp \) violates clause 3, since the derivative of the source variable appears in the body of \( cp \), which is not non-expansive.

14. Formats highlights

As in the case of approximated bisimulations, given that the kernel of bisimilarity metric coincides with probabilistic bisimilarity, the non-expansiveness, non-expansiveness, and Lipschitz formats discussed in this part are all obtained as refinements of the PGSOS format. Specifically, since the quantitative behaviour of processes is all encoded in the distribution terms, the three formats impose constraints on the form of the targets of PGSOS rules.

In the non-expansive format (Definition 45) only convex combination of variables and constants is admitted as targets. Essentially, \( \text{trgt}(r) = \sum_{i \in \Theta} p_i \theta_i \), and in each distribution term \( \theta_i \), we have at most one occurrence of one source variable, or of one of its derivatives, and this occurrence cannot be in the context of any operator.

The non-expansive format (Definition 46) allows \( \text{trgt}(r) \) to be a convex combination of distribution terms in which also non-constant operators occur and are applied to source variables and their derivatives, provided that, for each source variable, at most one occurrence of that variable, or its derivatives, occurs. This means that, in each \( \theta_i \), source variables, or their derivatives, can occur only in the scope of non-expansive operators that do not amplify their distance. However, all the variables occurring in \( \theta_i \) contribute to the overall distance.

The Lipschitz format (Definition 47) is more permissive, but it requires to inspect the PGSOS rules operator-by-operator. Given an operator \( f \), we use two natural numbers, \( n_f \) and \( m_f \), as upper bounds to the number of occurrences in the targets of the rules for
\( f \) of, respectively, source variables and their derivatives, and the nested operators in which a source variable can occur. Moreover, the occurrences of derivatives of source variables can only be in the scope of non-expansive operators. These constraints ensure that the distance between the instances of the arguments of operator \( f \) is expanded at most a factor \( p_j \cdot \lambda \cdot n_j \cdot L_{\theta_j} \), where \( L_{\theta_j} \) is the maximum of the Lipschitz factor of the \( m_j \) nested operators.

### Part 5. Concluding remarks

We have collected and discussed the formats for probabilistic behavioural equivalences, approximated equivalences, and behavioural metrics, over generative, reactive, and nondeterministic probabilistic transition systems, that have been proposed in the literature. Summarising, we have presented:

- The GTSS format ensuring the congruence property of generative bisimilarity over GPTs (Section 7.1).
- The RTSS format ensuring the congruence property of reactive bisimilarity over RTTs (Section 7.2).
- The PGSOS format ensuring the congruence property of probabilistic bisimilarity, the precongruence property of probabilistic ready similarity, and, in the positive version, the precongruence property of probabilistic similarity over NPTTs (Section 8.1).
- The PRBB-safe format ensuring the congruence property of rooted branching bisimilarity, and the PBB-safe format for that of branching bisimilarity over NPTTs (Section 8.2).
- The agb-safe format ensuring the non-expansiveness property of generative \( \epsilon \)-bisimulations over GPTSs (Section 10.1).
- The arb-safe format ensuring the non-expansiveness property for reactive \( \epsilon \)-bisimulations over RTTs (Section 10.2).
- The \( \epsilon \)-PGSOS format ensuring the non-expansiveness property for \( \epsilon \)-bisimulations over NPTTs (Section 11).
- The non-extensiveness, non-expansiveness, and Lipschitz continuity formats ensuring, respectively, the non-extensiveness, non-expansiveness and Lipschitz continuity of bisimilarity metric over NPTTs (Section 13).

To show that none of the formats we discussed is too restrictive, in Tables 11 and 12 we report the list of the formats that are satisfied by the principal operators defined in probabilistic process algebras. In Table 11 we consider the operators defined in Tables 3–5, and the formats for generative and reactive (approximated) bisimulations. Likewise, in Table 12 we list the formats for (bi)simulations, \( \epsilon \)-bisimulations and bisimulations metrics over NPTTs satisfied by the operators presented in Table 2. We remark that the unary operators of relabelling, restriction, and priority violate the non-extensiveness format for bisimilarity metrics, as indicated in Table 12. However, since they do satisfy the non-expansiveness format and they are unary, they show a non-extensive behaviour (as, in their case, the sum of the distances over the arguments always coincides with the maximum of those distances).

We now conclude our survey with a brief review of other probabilistic models existing in the literature, but for which no format has been proposed, followed by an informal presentation of formats proposed outside of the process algebraic setting.

### 15. Other models

In this survey, we focused our attention on three specific extensions of LTSs to the probabilistic setting: the generative, the reactive, and the nondeterministic and probabilistic models. The rationale behind this choice is quite straightforward: to the best of our knowledge these are the only models for which specification formats have been given. Moreover, we wanted to support compositional reasoning over systems in which nondeterministic aspects of the behaviour are combined with probabilistic ones. Each interpretation of this combination led to a different model, and consequently to a different format (as perfectly exemplified by the studies on strong probabilistic bisimilarity in Sections 7 and 8).

However, in the literature we can find other proposals of semantics models for probabilistic processes, for which no rule format has been given. For sake of completeness, below we give an informal presentation of those models. We remark that all the models discussed below are semistochastic.
Stratified model. The stratified model [72] captures the branching structure of purely probabilistic choices and follows a bisemantical approach to operational semantics. It is characterised by two types of transitions: the action transitions and the probability transitions. The former are of the form \( i \rightarrow u \) as in the classic LTSs. The latter ones allow for expressing explicitly probabilistic choices, and are of the form \( i \rightarrow \lambda < i \rightarrow p \rightarrow \mu \). (The role of the index \( i \) is similar to that of indexes in the generative and reactive models.) The operational rules are defined in such a way that processes can perform either only action transitions, or only probability transitions, and are therefore divided into action processes and probability processes, respectively. We remark that action processes have (at most) one outgoing action transition, the choice in this model is purely probabilistic, and there is no construct allowing for nondeterministic behaviour.

Alternating model. The alternating model stems from [115] and it extends the stratified model with nondeterministic choices, allowing for both internal and external nondeterminism.

In this setting, processes are partitioned into nondeterministic processes and probabilistic processes. The former ones can perform only (nondeterministic) action transitions; the latter ones only probability transitions. In particular, each probability transition is defined between a probabilistic process and a nondeterministic one. (Notice that this is another difference with the stratified model, since there probability transitions may end in probability processes.) Conversely, no constraint is imposed, in general, on the target of nondeterministic action transitions. In case these transitions are required to end in a probabilistic process, the model is then called strictly alternating (also known as Labelled Concurrent Markov Chains [76]).

Although the strictly alternating model may resemble the NPTS model, we remark that:

- in NPTSs there is no partition over the set of processes, and
- the probabilistic transitions that may be induced from the probability distributions do not have processes as their targets, but a discrete measure over them (the Dirac distribution).

Given the absence of a partition over processes, the NPTS model is also referred to as the non-alternating model, to mark the difference with the alternating model [104].

Markov models. Markov models are used to capture the semantics of those processes (and systems) whose probabilistic behaviour can be expressed in terms of a Markovian process (in the purely mathematical sense of the term). In the literature, we can find a wealth of proposals of such models that differ for the initial knowledge on the state space, the topology of the latter, the possibility to interact with the environment (and, thus, the presence of nondeterminism), whether timed aspects of computation are taken into account and how, etc. Just to make a few examples, the most common Markov models used in the literature are Markov Chains (and their labelled, continuous/discrete-time, continuous/discrete-space, variations), Labelled Markov Processes, Markov Decision Processes, and Hidden Markov models.

We spare the reader from a detailed presentation of these models, their variations, and all the other models not mentioned above, as this is out of the scope of our work. We limit ourselves to point out the principal difference between Markov models and the probabilistic models discussed so far: the relation between processes and probability. Markov models are characterised by either a transition probability function that maps a process (or a pair process, label) onto a distribution over processes, or a transition matrix that maps a pair of processes (or a triple process, label, process) onto a probability weight. These transition functions play the role of the Markov kernel of the Markovian process that represents the behaviour of the considered system. We remark that, as in Markovian processes, this means that the probability of reaching a particular process (or set of processes) depends only on the current process. In the various probabilistic ex-
tensions of LTSs given above, the labelled transitions are defined as relations either between two processes, or between a process and a distribution. In particular, the probability weight that is used as label of the transition it is not obtained, at least in principle, from the process performing that transition. This holds true also for the NPTS model, in which, technically speaking, the distribution reached via a transition does not depend on the process performing that transition, but on the resolution of nondeterministic choices.

16. The meta-model generalisation: a format for ULTraS

The realm of quantitative aspects of systems behaviour is much more various and complex than the distinction in the models considered above. This is due not only to possibly different interpretations of the nondeterminism-probability interplay, but also (and mainly) to the use of quantitative information other than probability. For instance, we might want to consider the execution rates of computation steps [14,85,99], the concentration of substances in biological/chemical systems [38,111,113], the load on the nodes in a network [68], or any other information of the performance of a system [60,78], etc. In the literature, we can find a wealth of semantic models and process calculi designed to capture a specific quantitative feature of systems behaviour. Despite the intuitive advantages of this richness, the drawback of their peculiarity is that the tools and formal results developed for one calculus cannot be used to reason on the others.

This inconvenience led to the search for meta-models, namely semantic frameworks that are general enough to offer uniform definitions and results that can be then instantiated and applied to a particular calculus or model. The most prominent examples of meta-models in the quantitative setting are Functional Transition Systems [89], Weighted Labelled Transition Systems [56,84], and Uniform Labelled Transition Systems (ULTraS) [23]. As our aim with this survey is to cover specification formats for probabilistic systems, we do not discuss these meta-models and the results obtained on them in detail. Nonetheless, given the generality of the ULTraS model, and for sake of completeness, we give a bird’s-eye view of the Weight Function GSOS (WF-GSOS) format proposed in [94] for the syntactic presentation of ULTraS.

Transitions in the ULTraS model are a generalisation of the transition relation presented in Section 3.3 for the NPTS model (Definition 8): each state can select nondeterministically an action-labelled transition leading to a state reachability weight function, i.e., any function quantifying the likelihood of a state to be reached in that step. Following [94], transitions are then of the form \( x \xrightarrow{a} \rho \) where \( x \) is a state, \( a \) is an action, and \( \rho \) is a weight function assigning to each state a value from a given commutative monoid \( \mathfrak{B} \).

Definition 49 (\( \mathfrak{B} \)-ULTraS, [94, Definition 2.1]). Given a commutative monoid \( \mathfrak{B} = (\mathbb{W}, +, 0) \), a \( \mathfrak{B} \)-weighted ULTraS is a triple \((X, A, \rightarrow)\) where: 1. \( X \) is a set of states; 2. \( A \) is a set of labels (actions); 3. \( \rightarrow \subseteq X \times A \times \{ X \rightarrow W \} \) is a transition relation where \( [X \rightarrow W] \) denotes the set of all weight functions from \( X \) to the carrier set of \( \mathfrak{B} \). A \( \mathfrak{B} \)-ULTraS is image finite iff for any \( x \in X \) and \( a \in A \) the set \( \{ \rho | x \xrightarrow{a} \rho \} \) is finite and all the weight functions \( \rho \) in it have a finite support.

The intuition behind the notion of bisimulation over \( \mathfrak{B} \)-ULTraS states is also a generalisation of that over NPTSs: two states are bisimilar if by performing the same actions they reach weight functions that assign the same weights to bisimulation classes of states. Given \( C \subseteq X \), the total weight assigned to \( C \) by \( \rho \) is defined by \( \| \rho \|_C = \sum_{x \in C} \rho(x) \). Then, given any relation \( R \) between two sets \( X \) and \( Y \), and its subset closure \( R^* \), the lifting \( R_{\mathfrak{B}} \) over weight functions is defined as \( (\rho_1, \rho_2) \in R_{\mathfrak{B}} \) iff for all \( (C, D) \in R^* \) it holds that \( \| \rho_1 \|_C = \| \rho_2 \|_D \).

Definition 50 (Bisimulation, [94, Definition 2.6]). Let \((X, A, \xrightarrow{x})\) and \((Y, A, \xrightarrow{y})\) be two image finite \( \mathfrak{B} \)-ULTraS. A relation \( R \subseteq X \times Y \) is a bisimulation iff \((x, y) \in R\) implies that for each \( a \in A \):

- if \( x \xrightarrow{a} x \rho_1 \), then there exists \( y \xrightarrow{a} y \rho_2 \) such that \((\rho_1, \rho_2) \in R_{\mathfrak{B}}\);
- if \( y \xrightarrow{a} y \rho_2 \), then there exists \( x \xrightarrow{a} x \rho_1 \) such that \((\rho_1, \rho_2) \in R_{\mathfrak{B}}\).

States \( x, y \) are said to be bisimilar if there is a bisimulation relation \( R \) such that \((x, y) \in R \).

The WF-GSOS format is then obtained as an extension of the classic GSOS format. A new signature, called the weight signature, is introduced for a syntactic representation of weight functions, in a similar fashion as distribution terms are used as the syntactic counterpart of probability distributions (see Section 4.1).

Definition 51. [WF-GSOS rule, [94, Definition 3.1]] Let \( \Sigma \) and \( \Theta \) be the process signature and the weight signature respectively. A WF-GSOS rule over them is a rule of the form

\[
\begin{aligned}
\{ x_i \xrightarrow{\phi_{ij}} \} \quad \text{if} \quad \Sigma \cup \text{wfr} \cup \Theta \\
\{ x_i \xrightarrow{\phi_{ik}} \} \quad \text{if} \quad \Sigma \cup \text{wfr} \\
\{ y_k \} \quad \text{if} \quad \Theta \\
\{ y_k \} \quad \text{if} \quad \Theta \\
\end{aligned}
\]

where

\[
f(x_1, \ldots, x_n) \xrightarrow{\psi} y
\]
A WF-GSOS rule is triggered by a tuple \( \langle C_1, \ldots, C_n \rangle \) of enabled labels and by a tuple \( \langle v_1, \ldots, v_p \rangle \) of weights iff \( A_i \subseteq C_i, B_i \cap C_i = \emptyset, \) and \( w_j = v_j \) for \( 1 \leq i \leq n \) and \( 1 \leq j \leq p. \)

Weight terms are evaluated to weight functions by means of interpretations \( \| \cdot \| \) allowing each term to represent finitely many weight functions.

**Definition 52 (WF-GSOS specification, \cite[Definition 3.4]{Andova2003}).** An image finite WF-GSOS specification over \( \mathfrak{B}, A, \Sigma, \) and \( \Theta \) is a pair \((\mathfrak{R}, \| \cdot \|)\) where \( \| \cdot \| \) is a weight term interpretation and \( \mathfrak{R} \) is a set of rules complaint with Definition 51 and such that only finitely many rules share the same operator source \( f, \) the same label \( c \) in the conclusion, and the same trigger \( \langle A_1, \ldots, A_n \rangle \langle \langle v_1, \ldots, v_p \rangle \).

**Theorem 12 (Congruence, \cite[Theorem 3.6]{Andova2003}).** The bisimulation on the ULPraS induced by a WF-GSOS specification is a congruence with respect to the process signature.

Alongside this result, in \cite{Andova2003} the authors provide a categorical presentation of their WF-GSOS format (also discussed in a previous work by the same authors \cite{Andova2001}). This co-algebraic characterisation of the WF-GSOS format is a generalisation of the W-GSOS format from \cite{Aceto2000}, i.e., a format for weighted transition systems over a commutative monoid \( W. \) In particular, the Stochastic GSOS (SGSOS) format is presented as a special case of the W-GSOS format ensuring that stochastic bisimilarity \cite{Andova2000} is a congruence. Related to this line of research (which is outside of the scope of our survey), in \cite{Baier2004, Baier2005} the bialgebraic approach \cite{Baier2006} is used to obtain rule formats over probabilistic and stochastic systems with continuous states.

**Declaration of competing interest**

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Valentina Castiglioni reports financial support was provided by The Icelandic Centre for Research.

**Data availability**

No data was used for the research described in the article.

**References**