A Formalized Hierarchy of Probabilistic System Types
Proof Pearl

Johannes Hölzl[1], Andreas Lochbihler[2], and Dmitriy Traytel[1]

[1] Fakultät für Informatik, Technische Universität München, Germany
[2] Institute of Information Security, Department of Computer Science, ETH Zurich, Switzerland

Abstract. Numerous models of probabilistic systems are studied in the literature. Coalgebra has been used to classify them into system types and compare their expressiveness. In this work, we formalize the resulting hierarchy of probabilistic system types in Isabelle/HOL by modeling the semantics of the different systems as codatatypes. This approach yields simple and concise proofs, as bisimilarity coincides with equality for codatatypes. On the way, we develop libraries of bounded sets and discrete probability distributions and integrate them with the facility for (co)datatype definitions.

1 Introduction

The framework of coalgebra provides a unified view on various ways of modeling (probabilistic) systems [2, 20, 21, 24]. A system is represented as a function of type $\sigma \Rightarrow \sigma\ \mathsf{F}$ that describes the possible evolutions of a state of type σ. Here, the functor F (written postfix) determines the type of the system. For example, a non-deterministic labeled transition system corresponds to a function $\sigma \Rightarrow (\alpha \Rightarrow \sigma\ \mathsf{set})$, which returns the set of the possible successor states for each label of type α. Similarly, a Markov chain can be characterized by a function from a state to the probability distribution over the successor states. More complicated types combine non-deterministic and probabilistic aspects in different ways.

Bartels et al. [2] and Sokolova [21] compare the expressiveness of system types found in the literature and arrange them in a hierarchy. They define a type of systems to be at least as expressive as another if every system of the latter can be mapped to a system of the former such that the mapping preserves and reflects bisimilarity, where two systems are bisimilar iff they cannot be distinguished by finite observations [17].

In this paper, we formalize the probabilistic system types (§5) and their hierarchy (§6) in Isabelle/HOL. The salient feature is that we model the system types as codatatypes (§2) rather than functions as done in the original proofs [21]. On codatatypes, bisimilarity coincides with equality, which allows for convenient equational reasoning. This makes the proofs simple and concise, i.e., highly automated and without lots of technical clutter. Our formalization is publicly available [14].

To construct the codatatypes, we introduce new types to express non-deterministic and probabilistic choice, namely bounded (non-empty) powerset (§3) and discrete probability distributions (§4). We integrate them with Isabelle's new package for (co)datatypes [4, 6, 22]. Thus, we can define the codatatypes directly, which demonstrates the versatility of the new package. Moreover, future formalizations [18, 25] benefit, too, as recursion in (co)datatypes may now occur under discrete distributions and bounded sets.

Our work is more than just an exercise in formalization. We extend the original hierarchy with additional standard system types and discover new interconnections by considering systems extended with an additional label (§6.3). Moreover, the formalization has revealed a flaw in the original hierarchy proof. We show that Vardi systems (also known as concurrent Markov chains [23]) do not satisfy the assumptions required in [21] and therefore must be (partially) dismissed from the hierarchy (§6.4).

2 Preliminaries: Codatatypes via Bounded Natural Functors

The flexibility of Isabelle's (co)datatype package originates from a semantic criterion that defines where (co)recursion may appear on the right-hand side of a (co)datatype declaration (in contrast to syntactic criteria employed by most if not all other proof assistants including past versions of Isabelle).

The core of the semantic criterion relies on the notion of a *bounded natural functor* (BNF) [4, 22]. Here, we shortly introduce BNFs targeted at our application. A (unary) BNF is a type expression $\alpha\ \mathsf{F}$ with a type parameter α equipped with a polymorphic map function $\mathsf{map}_\mathsf{F} :: (\alpha \Rightarrow \beta) \Rightarrow \alpha\ \mathsf{F} \Rightarrow \beta\ \mathsf{F}$, a polymorphic set function $\mathsf{set}_\mathsf{F} :: \alpha\ \mathsf{F} \Rightarrow \alpha\ \mathsf{set}$, and an infinite cardinal bound bd_F on the number of elements returned by set_F. Additionally, these constants must satisfy certain properties (e.g., map_F is *functorial*, i.e., preserves identities and composition, and set_F is a *natural transformation*, i.e., commutes with map_F). For example, the type of (finite) lists $\alpha\ \mathsf{list}$ forms a BNF with the standard map function map and the function set returning the set of the list's elements.

The semantic criterion allows (co)recursion to occur nested under BNFs. For example, the (co)datatypes $\alpha\ \mathsf{tree}$ and $\alpha\ \mathsf{ltree}$ of finitely branching trees nest the (co)recursive occurrences of $\alpha\ \mathsf{tree}$ and $\alpha\ \mathsf{ltree}$ under the BNF list:

datatype $\alpha\ \mathsf{tree} = \mathsf{Node}\ \alpha\ (\alpha\ \mathsf{tree}\ \mathsf{list})$
codatatype $\alpha\ \mathsf{ltree} = \mathsf{Node}\ \alpha\ (\alpha\ \mathsf{ltree}\ \mathsf{list})$

While only trees of finite depth inhabit the datatype $\alpha\ \mathsf{tree}$, the codatatype $\alpha\ \mathsf{ltree}$ also hosts trees of infinite depth. For example, the full binary tree z containing 0 everywhere is defined by primitive corecursion [4]: **primcorec** $\mathsf{z} :: \mathsf{nat}\ \mathsf{ltree}$ **where** $\mathsf{z} = \mathsf{Node}\ 0\ [\mathsf{z}, \mathsf{z}]$.

Users can register custom types as BNFs by supplying the required constants and discharging the proof obligations for the BNF properties. Newly registered BNFs can then participate in further (co)datatype declarations. For example, after registering Isabelle's type of finite sets $\alpha\ \mathsf{fset}$ as a BNF, we can define unordered finitely branching trees of potentially infinite depth: **codatatype** $\alpha\ \mathsf{lftree} = \mathsf{Node}\ \alpha\ (\alpha\ \mathsf{lftree}\ \mathsf{fset})$.

In general, BNFs can have arbitrary arity and may depend on additional *dead* type variables that are ignored by the map function. For example, the sum and product types are binary BNFs, while the function type $\alpha \Rightarrow \beta$ is a unary BNF with the dead variable α (BNFs thereby disallow recursion through negative positions [10]). Compositions of BNF are again BNFs. We say that a BNF $\alpha\ \mathsf{F}$ *induces* a codatatype C_F

codatatype $\mathsf{C}_\mathsf{F} = \mathsf{Ctr}_\mathsf{F}\ (\mathsf{C}_\mathsf{F}\ \mathsf{F})$

with a single bijective *constructor* $\mathsf{Ctr}_\mathsf{F} :: \mathsf{C}_\mathsf{F}\ \mathsf{F} \Rightarrow \mathsf{C}_\mathsf{F}$, its inverse *destructor* $\mathsf{Dtr}_\mathsf{F} :: \mathsf{C}_\mathsf{F} \Rightarrow \mathsf{C}_\mathsf{F}\ \mathsf{F}$ and the associated *coiterator* $\mathsf{unfold}_\mathsf{F}$ defined by primitive corecursion:

primcorec $\mathsf{unfold_F} :: (\alpha \Rightarrow \alpha\ \mathsf{F}) \Rightarrow \alpha \Rightarrow \mathsf{C_F}$ **where**
$\mathsf{Dtr_F}\ (\mathsf{unfold_F}\ s\ a) = \mathsf{map_F}\ (\mathsf{unfold_F}\ s)\ (s\ a)$

Finally, induced codatatypes are equipped with a *coinduction* rule for proving equality by exhibiting a *bisimulation relation witness R*:

$$\frac{R\ x\ y \quad \forall x\ y.\ R\ x\ y \longrightarrow \mathsf{rel_F}\ R\ (\mathsf{Dtr_F}\ x)\ (\mathsf{Dtr_F}\ y)}{(x :: \mathsf{C_F}) = (y :: \mathsf{C_F})}$$

where the *relator* $\mathsf{rel_F} :: (\alpha \Rightarrow \beta \Rightarrow \mathsf{bool}) \Rightarrow \alpha\ \mathsf{F} \Rightarrow \beta\ \mathsf{F} \Rightarrow \mathsf{bool}$ lifts binary relations over elements to binary relations over the functor F. It is defined for each BNF canonically in terms of $\mathsf{map_F}$ and $\mathsf{set_F}$ (where π_1 and π_2 denote the standard product projections):

$$\mathsf{rel_F}\ R\ x\ y = \exists z.\ \mathsf{set_F}\ z \subseteq \{(x, y) \mid R\ x\ y\} \wedge \mathsf{map_F}\ \pi_1\ z = x \wedge \mathsf{map_F}\ \pi_2\ z = y \qquad (1)$$

3 Bounded Powerset

In this and the next section we define three new types and register them as BNFs. We start with the simpler two: bounded sets and non-empty bounded sets, with which we will model non-determinism on a state space. Our new type and its BNF structure generalize the existing BNFs for finite sets α fset and countable sets α cset in Isabelle/HOL's library. Note that Isabelle's standard type of unbounded sets α set is not a BNF, due to the absence of a cardinal bound on the number of elements contained in a set.

As for bounded sets, we cannot directly express the dependence of a type on a cardinal bound constant within the simply typed logic of Isabelle. A standard trick [11] is to let the type depend on a type κ (and thereby on κ's cardinality) instead. We obtain the following type definitions for the type $\alpha\ \mathsf{set}^\kappa$ of strictly κ-bounded sets:

typedef $\alpha\ \mathsf{set}^\kappa = \{A :: \alpha\ \mathsf{set} \mid |A| <_o |\mathsf{UNIV} :: \kappa\ \mathsf{set}| +_c \aleph_0\}$

The operators $|-|$, $<_o$, $+_c$ and the constant $\aleph_0 = |\mathsf{UNIV} :: \mathsf{nat\ set}|$ are part of Isabelle's library of cardinals [5]—their exact definition is irrelevant; they encode the intuition that $\alpha\ \mathsf{set}^\kappa$ contains all sets of strictly smaller cardinality than κ if κ is an infinite type (in which case $|\mathsf{UNIV} :: \kappa\ \mathsf{set}| +_c \aleph_0 =_o |\mathsf{UNIV} :: \kappa\ \mathsf{set}|$) and all finite sets otherwise (since $|\mathsf{UNIV} :: \kappa\ \mathsf{set}| +_c \aleph_0 =_o \aleph_0$ for finite κ). In other words: If we instantiate κ with a finite or countable type, then $\alpha\ \mathsf{set}^\kappa$ is isomorphic to α fset, and if we instantiate κ with the cardinal successor of $\aleph_0$ [5], then $\alpha\ \mathsf{set}^\kappa$ is isomorphic to α cset.

It is easy to define the map and set function for $\alpha\ \mathsf{set}^\kappa$ using the Lifting tool [15]:

lift-definition $\mathsf{map_{set}} :: (\alpha \Rightarrow \beta) \Rightarrow \alpha\ \mathsf{set}^\kappa \Rightarrow \beta\ \mathsf{set}^\kappa$ **is** image
lift-definition $\mathsf{set_{set}} :: \alpha\ \mathsf{set}^\kappa \Rightarrow \alpha\ \mathsf{set}$ **is** id

The map function only acts on the element type α, which implies that κ will be a dead type variable of the following BNF structure. The bound for the set size in the above **typedef** command serves as bound for the BNF, too.

bnf $\alpha\ \mathsf{set}^\kappa$ map: $\mathsf{map_{set}}$ set: $\mathsf{set_{set}}$ bd: $|\mathsf{UNIV} :: \kappa\ \mathsf{set}| +_c \aleph_0$

To finish the registration of α set^κ as a BNF, the **bnf** command requires the user to discharge the following proof obligations. (The proofs of these properties are straightforward generalizations of the ones for α fset.)

$$\mathsf{map}_{\mathsf{set}}\ \mathsf{id} = \mathsf{id} \qquad \mathsf{map}_{\mathsf{set}}\ (f \circ g) = \mathsf{map}_{\mathsf{set}}\ f \circ \mathsf{map}_{\mathsf{set}}\ g$$

$$|\mathsf{set}_{\mathsf{set}}\ X| \leq_{\mathrm{o}} |\mathsf{UNIV} :: \kappa\ \mathsf{set}| +_{\mathrm{c}} \aleph_0 \qquad \mathsf{set}_{\mathsf{set}} \circ \mathsf{map}_{\mathsf{set}}\ f = \mathsf{image}\ f \circ \mathsf{set}_{\mathsf{set}}$$

$$(\forall x \in \mathsf{set}_{\mathsf{set}}\ X.\ f\,x = g\ x) \longrightarrow \mathsf{map}_{\mathsf{set}}\ f\ X = \mathsf{map}_{\mathsf{set}}\ g\ X$$

$$\mathsf{rel}_{\mathsf{set}}\ R \bigcirc\!\!\bigcirc \mathsf{rel}_{\mathsf{set}}\ S \sqsubseteq \mathsf{rel}_{\mathsf{set}}\ (R \bigcirc\!\!\bigcirc S)$$

The first five being easy to discharge, the last proof obligation requires some explanation: $\sqsubseteq$ denotes implication lifted to binary predicates and $\bigcirc\!\!\bigcirc$ denotes the relational composition of binary predicates. With this definition the last proof obligation is equivalent to what in categorical jargon is called *weak pullback preservation*. We can show that bounded sets preserve weak pullbacks iff the bound on the number of elements is infinite or ≤ 2. In our case, the bound is infinite due to the addition of $\aleph_0$, therefore α set^κ is a BNF. This corrects an earlier claim that α set^κ is a BNF for all κ [22].

Similarly to α set^κ, we define (and prove being a BNF) the type α set_1^κ of nonempty strictly κ-bounded sets which will be used to model Markov decision processes.

typedef $\alpha\ \mathsf{set}_1^\kappa = \{A :: \alpha\ \mathsf{set} \mid A \neq \varnothing \wedge |A| <_{\mathrm{o}} |\mathsf{UNIV} :: \kappa\ \mathsf{set}| +_{\mathrm{c}} \aleph_0\}$

4 Probability Mass Functions

We introduce a type of *probability mass functions* (*pmf*) on a type α, representing distributions of discrete random variables on α. There are two views on a pmf: (1) as a non-negative real-valued function which sums up to 1, and (2) as a discrete probability measure which has a countable set S which has probability 1. Both views are available in our formalization. In this paper, however, we only present the measure view, as we lift all presented definitions from the existing formalization of measure theory [13].

A measure $M :: \alpha$ measure consists of a σ-algebra of measurable sets sets M and a measure function $\mu\ M$ that is non-negative and countably-additive on sets M. A probability distribution M assigns 1 to the whole space ($\mu\ M\ \mathsf{UNIV} = 1$). It is discrete iff every set is measurable (sets $M = \mathsf{UNIV}$) and there exists a countable set S with $\mu\ M\ S = 1$.

typedef $\alpha\ \mathsf{pmf} = \{M :: \alpha\ \mathsf{measure} \mid$
$\mu\ M\ \mathsf{UNIV} = 1 \wedge \mathsf{sets}\ M = \mathsf{UNIV} \wedge (\exists S.\ \mathsf{countable}\ S \wedge \mu\ M\ S = 1)\}$

The command **typedef** generates a representation function $\mathsf{measure}_{\mathsf{pmf}} :: \alpha\ \mathsf{pmf} \Rightarrow \alpha\ \mathsf{measure}$. By declaring it as a coercion function, we can omit it in most cases. In particular, we write $\mu\ p\ A$ for $\mu\ (\mathsf{measure}_{\mathsf{pmf}}\ p)\ A$. So, the probability mass of a value x is the measure of its singleton set $\{x\}$. We lift the support set from the measure definition:

lift-definition $\mathsf{set}_{\mathsf{pmf}} :: \alpha\ \mathsf{pmf} \Rightarrow \alpha\ \mathsf{set}$ **is** $\lambda M.\ \{x \mid \mu\ M\ \{x\} \neq 0\}$

Next, we lift the monadic operators $\mathsf{bind}_{\mathsf{pmf}}$ and $\mathsf{return}_{\mathsf{pmf}}$ from the Giry monad on measure spaces [8] to pmfs. The map function $\mathsf{map}_{\mathsf{pmf}}$ is then defined in a standard way as a combination of these monadic operators.

lift-definition $\mathsf{bind}_{\mathsf{pmf}} :: \alpha\ \mathsf{pmf} \Rightarrow (\alpha \Rightarrow \beta\ \mathsf{pmf}) \Rightarrow \beta\ \mathsf{pmf}$ **is** bind

lift-definition $\mathsf{return}_{\mathsf{pmf}} :: \alpha \Rightarrow \alpha\ \mathsf{pmf}$ **is** return (count-space UNIV)

definition $\mathsf{map}_{\mathsf{pmf}} :: (\alpha \Rightarrow \beta) \Rightarrow \alpha\ \mathsf{pmf} \Rightarrow \beta\ \mathsf{pmf}$ **where**
$\quad \mathsf{map}_{\mathsf{pmf}}\ f\ M = \mathsf{bind}_{\mathsf{pmf}}\ M\ (\lambda x.\ \mathsf{return}_{\mathsf{pmf}}\ (f\ x))$

When working with general measure spaces, all functions must be shown to be measurable. In our restricted discrete setting all function are trivially measurable, hence characteristic theorems about $\mathsf{bind}_{\mathsf{pmf}}$ and $\mathsf{return}_{\mathsf{pmf}}$ carry no measurability assumptions:

$\mathsf{bind}_{\mathsf{pmf}}\ (\mathsf{bind}_{\mathsf{pmf}}\ M\ f)\ g = \mathsf{bind}_{\mathsf{pmf}}\ M\ (\lambda x.\ \mathsf{bind}_{\mathsf{pmf}}\ (f\ x)\ g)$
$\mathsf{bind}_{\mathsf{pmf}}\ (\mathsf{return}_{\mathsf{pmf}}\ x)\ f = f\ x$
$\mathsf{bind}_{\mathsf{pmf}}\ M\ \mathsf{return}_{\mathsf{pmf}} = M$

The behavior of $\mathsf{bind}_{\mathsf{pmf}}$ and $\mathsf{return}_{\mathsf{pmf}}$ under $\mathsf{set}_{\mathsf{pmf}}$ is as expected:

$\mathsf{set}_{\mathsf{pmf}}\ (\mathsf{bind}_{\mathsf{pmf}}\ M\ f) = \bigcup_{x \in \mathsf{set}_{\mathsf{pmf}}\ M} \mathsf{set}_{\mathsf{pmf}}\ (f\ x)$
$\mathsf{set}_{\mathsf{pmf}}\ (\mathsf{return}_{\mathsf{pmf}}\ x) = \{x\}$
$(\forall x \in \mathsf{set}_{\mathsf{pmf}}\ M.\ f\ x = g\ x) \longrightarrow \mathsf{bind}_{\mathsf{pmf}}\ M\ f = \mathsf{bind}_{\mathsf{pmf}}\ M\ g$

Another standard construction in probability theory is the conditional probability $\Pr(X \in A \mid X \in B) = \Pr(X \in A \wedge X \in B) / \Pr(X \in B)$, i.e. the probability that the random variable X has a result in A under the assumption that X is in B. This requires that X being in B has positive probability. In Isabelle's measure theory, the function uniform-measure expresses a conditional probability. It returns a probability space when the measure of the set B is positive. Clearly, lifting uniform-measure to pmfs works only if we restrict B to such sets. Therefore, we fix a pmf p and a set B with the assumption $\mathsf{set}_{\mathsf{pmf}}\ p \cap B \neq \varnothing$, which is equivalent to $\mu\ p\ B \neq 0$.

lift-definition $\mathsf{cond}_{\mathsf{pmf}} :: \alpha\ \mathsf{pmf}$ **is** $\mathsf{uniform\text{-}measure}\ (\mathsf{measure}_{\mathsf{pmf}}\ p)\ B$

Whenever $\mathsf{set}_{\mathsf{pmf}}\ p \cap B \neq \varnothing$ holds we will from now on write $\mathsf{cond}_{\mathsf{pmf}}\ p\ B$. We then have $\mu\ (\mathsf{cond}_{\mathsf{pmf}}\ p\ B)\ A = \mu\ p\ (A \cap B)\ /\ \mu\ p\ B$ and hence $\mathsf{set}_{\mathsf{pmf}}\ (\mathsf{cond}_{\mathsf{pmf}}\ p\ B) = \mathsf{set}_{\mathsf{pmf}}\ p \cap B$.

Probability Mass Functions as a BNF We now prove that α pmf is a BNF such that the codatatypes for the probabilistic systems can recurse through α pmf. To that end, we define the relator $\mathsf{rel}_{\mathsf{pmf}}$ on pmfs and prove that $\mathsf{set}_{\mathsf{pmf}}$, $\mathsf{map}_{\mathsf{pmf}}$, and $\mathsf{rel}_{\mathsf{pmf}}$ satisfy the BNF properties. The definition of $\mathsf{rel}_{\mathsf{pmf}}\ R\ p\ q$ is canonical as in (1). The existentially quantified z corresponds to a matrix of non-negative reals with a row and a column for each element in the support of p and q, respectively, such that (i) summing over a row i or a column j yields the mass of p or q concentrated in i or j, and (ii) positive entries are only at cells (i, j) for which $R\ i\ j$ holds. We call such a matrix an R-lifting matrix for p and q.

With the lemmas about $\mathsf{bind}_{\mathsf{pmf}}$, $\mathsf{return}_{\mathsf{pmf}}$, $\mathsf{set}_{\mathsf{pmf}}$ and the definition of $\mathsf{map}_{\mathsf{pmf}}$ we immediately derive the functorial BNF properties for pmfs with the cardinal bound $\aleph_0$. Only distributivity with composition has interesting proof, i.e., $\mathsf{rel}_{\mathsf{pmf}}\ R \circ\circ \mathsf{rel}_{\mathsf{pmf}}\ S \sqsubseteq \mathsf{rel}_{\mathsf{pmf}}\ (R \circ\circ S)$. That is, given an R-lifting matrix z_1 for pmfs p and q and an S-lifting matrix z_2 for q and r, we have to construct an $(R \circ\circ S)$-lifting matrix z for p and r. In the course of this work, we have formalized a series of three different constructions for z, each of which made the previous proof simpler and more concise. The steps are recorded in the changesets (mentioned below) of the Isabelle repository at `http:`

`//isabelle.in.tum.de/repos/isabelle`. This process illustrates how pmfs provide abstraction and lead to shorter proofs.

Initially, we followed Sokolova's construction [21]. She defines the matrix z as the sum of matrices z^j over $j \in \mathsf{set_{pmf}}\ q$ where each z^j is an T^j-lifting matrix for the jth column of z_1 and the jth row of z_2 (we ignore that the rows and columns do not sum to 1) where $T^j\ i\ k = R\ i\ j \wedge S\ \ j\ k$. An iterative algorithm constructs the matrix z^j by walking on a path from the upper left corner to the lower right and setting each entry to the maximum such that neither the row sum nor the column sum is exceeded. If the row sum is matched after setting the entry, the path continues down, if the column sum is matched, it goes to the left, if both are matched, it goes diagonally to the right and down. Her proof that z is an $(R \circ\circ S)$-lifting matrix for p and r requires five pages on paper [21, Lemmas 3.5.5, 3.5.6]. Our HOL formalization of a recursive version of the algorithm and the proof of the distributivity property is arduous and takes 577 lines (`4999a616336c`). By switching from real-valued functions to pmfs and using $\mathsf{map_{pmf}}$ in the construction of z from the z^j, we were able to shorten the proof to 406 lines (`43e07797269b`). Still, most of the proof script dealt with showing the equality of different summations.

Next, we realized that taking a path through the matrix and setting the entries to maximum values was needlessly convoluted. Instead, we fill the ith row of z^j by distributing z_1's value at (i, j) over the columns according to the jth row of z_2. This eliminates all the inductions and several bijections between the support sets and natural numbers, which were needed for the recursion. This is the proof by Jonsson et al. [16] formalized in 101 lines (`922d31f5c3f5`, `922d31f5c3f5`). Zanella [26] has previously formalized this proof for CertiCrypt using Audebaud's and Mohring's library [1]. His proof script needs 337 lines of Coq.

Finally, we noted that the distribution over the columns and the summation over the z^j yields a conditional probability. So, we now define z simply as

$$z = \mathsf{bind_{pmf}}\, z_1\ (\lambda(i, j).\ \mathsf{bind_{pmf}}\ (\mathsf{cond_{pmf}}\, z_2\ \{(j', k) \mid j' = j\})\ (\lambda(\_, k).\ \mathsf{return_{pmf}}(i, k)))$$

Thus, only one conjunct is shown with summations, namely of $\mathsf{map_{pmf}}\, \pi_1\ z = p$. The others are discharged by reasoning with the laws about $\mathsf{set_{pmf}}$, $\mathsf{bind_{pmf}}$, $\mathsf{cond_{pmf}}$, and $\mathsf{return_{pmf}}$. The following law is particularly useful. It generalizes the *law of total probability*, which states $\Pr(A) = \sum_n \Pr(A \mid B_n) \cdot \Pr(B_n)$ for a countably indexed partition B. Note that $\mathsf{bind_{pmf}}$ expresses the sum and R relates the events of a and b.

$$\frac{\begin{array}{c}\mathsf{rel_{set}}\ R\ (\mathsf{set_{pmf}}\ a)\ (\mathsf{set_{pmf}}\ b) \\ \forall x \in \mathsf{set_{pmf}}\ a.\ \forall y \in \mathsf{set_{pmf}}\ b.\ R\ x\ y \longrightarrow \mu\ a\ \{x \mid R\ x\ y\} = \mu\ b\ \{y \mid R\ x\ y\}\end{array}}{\mathsf{bind_{pmf}}\ b\ (\lambda y.\ \mathsf{cond_{pmf}}\ a\ \{x \mid R\ x\ y\}) = a} \qquad (2)$$

Here, the set relator $\mathsf{rel_{set}}\ R\ A\ B$ denotes $(\forall a \in A.\ \exists b \in B.\ R\ a\ b) \wedge (\forall b \in B.\ \exists a \in A.\ R\ a\ b)$. (Using the same notation for bounded and unbounded sets, this characterization also holds for the relator of bounded sets.) We use this law to show $\mathsf{map_{pmf}}\, \pi_2\ z = r$. Observe that $\mathsf{map_{pmf}}\, \pi_2\ z = \mathsf{map_{pmf}}\, \pi_2\ (\mathsf{bind_{pmf}}\ q\ (\lambda y.\ \mathsf{cond_{pmf}}\ z_2\ \{(y', z) \mid y' = y\}))$. Applying the law to the right-hand side yields $\mathsf{map_{pmf}}\, \pi_2\ z_2$, which equals r by assumption.

In the end, our proof is just 46 lines, which includes 18 lines for the proof of equation 2 (`224741ede5ae`). This confirms in our eyes that our pmf library is well designed. Also, we argue that the proof has gained in clarity from the reduction in size. We eliminated most of the technical transformations of sums and express them more abstractly.

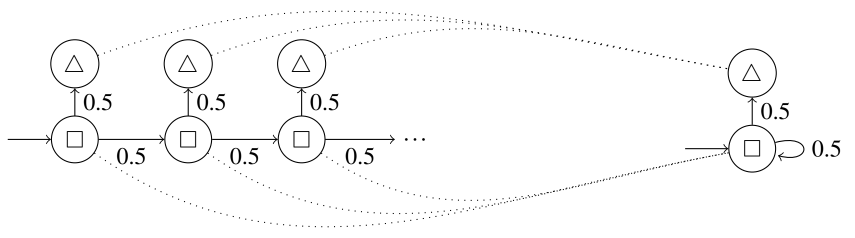


Fig. 1: Two labeled Markov chains (the dotted lines represent a bisimulation relation)

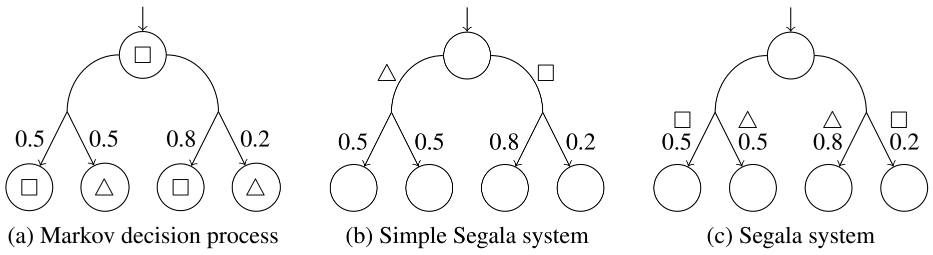


Fig. 2: Three probabilistic automata with non-deterministic and probabilistic choice

5 Probabilistic Systems

Probabilistic Systems as Probabilistic Automata First, we review the approach of modeling probabilistic systems as probabilistic automata. These automata fall into different classes depending on whether they make use of probabilistic and non-deterministic choice, where labels are placed, and whether transitions generate output for the environment or receive input from it.

Labeled Markov chains are a very simple class of probabilistic automata. Here each state has a label and specifies a probability distribution over the successor states. Figure 1 shows two labeled Markov chains. In our figures, □ or △ denote labels and numbers between 0 and 1 denote probabilities. The Markov chain on the right stores in the state only whether the system has reached the label △. In contrast, the one on the left additionally records in its states how many steps have been taken to reach △.

When modeling systems with probabilistic automata, we usually care only about the labels, not the states. In that respect, both Markov chains produce the same observations with the same probabilities. Thus, it is sensible to consider the two chains as being equivalent. Bisimulation captures this equivalence by identifying states which cannot be distinguished by observing the labels in any behavior originating from these states. For labeled Markov chains, a bisimulation relation is an equivalence relation R on the states such that whenever s and t are related by R, then their labels are the same and for all equivalence classes C of R, the probabilities of going to C from s and t are the same.

In Figure 1, the dotted lines show a bisimulation relation between the two chains. We say that labeled Markov chains on the same state space are bisimilar if the initial states are related in some bisimulation relation.

Combining non-deterministic and probabilistic choice, we get more complicated models. Figure 2 shows three examples. In a Markov decision process, each state has a label, but it may choose non-deterministically a distribution of the successor states. Graphically, the transition edges split after having taken the non-deterministic choice. In a simple Segala system, the label is attached to the non-deterministic transitions rather than the states. So, the transition generates the label instead of the state. And in a Segala system, the label is attached to the probabilistic choice rather than the non-deterministic one. For these more complicated systems, the definition of bisimulation is analogous to Markov chains, but more involved.

Coalgebraic View on Probabilistic Systems Next, we switch perspective and outline the coalgebraic approach to modeling probabilistic systems [2,21,24]. We recollect the basic coalgebraic vocabulary (an in-depth introduction can be found elsewhere [20]) and show how these notions are reflected in Isabelle/HOL.

Given a functor F, an F-*coalgebra* is defined as a pair (A, s) with the carrier set A and the structural mapping $s : A \to \mathsf{F}\ A$. In our typed environment of HOL, we restrict our attention to bounded natural functors and require the carrier set of a coalgebra to be the universe of a certain type σ. Therefore, for us a coalgebra is just a function $s :: \sigma \Rightarrow \sigma\ \mathsf{F}$ for a BNF $\sigma\ \mathsf{F}$. Intuitively, a coalgebra $s :: \sigma \Rightarrow \sigma\ \mathsf{F}$ describes a transition system whose states are in σ and each state $x :: \sigma$ evolves into $s\ x :: \sigma\ \mathsf{F}$. For example, if $\sigma\ \mathsf{F} = \sigma\ \mathsf{pmf}$, then $s\ x :: \sigma\ \mathsf{pmf}$ is a discrete probability distribution over the next states and s taken as a whole denotes an unlabeled Markov chain.

Bisimilarity can be defined uniformly on coalgebras [12]: states x and y of two systems s_1 and s_2 are F-*bisimilar* (written $x\ {}^{s_1}{\sim}_{\mathsf{F}}^{s_2}\ y$) iff there exists a relation $R ::$ $\alpha \Rightarrow \beta \Rightarrow \mathsf{bool}$ (called bisimulation) that relates x and y and for all related pairs of states x and y their evolutions $s_1\ x$ and $s_2\ y$ are related by $\mathsf{rel}_{\mathsf{F}}\ R :: \alpha\ \mathsf{F} \Rightarrow \beta\ \mathsf{F} \Rightarrow \mathsf{bool}$. Formally:

$$x\ {}^{s_1}{\sim}_{\mathsf{F}}^{s_2}\ y = \exists R.\ R\ x\ y \wedge (\forall x\ y.\ R\ x\ y \longrightarrow \mathsf{rel}_{\mathsf{F}}\ R\ (s_1\ x)\ (s_2\ y))$$

It turns out that this generic notion coincides with the known concrete bisimilarity notions for all systems F that we consider [2,21]. We should note that for $\sigma\ \mathsf{F} = \sigma\ \mathsf{pmf}$ all states of all systems are bisimilar: $\forall s_1\ s_2\ x\ y.\ x\ {}^{s_1}{\sim}_{\mathsf{pmf}}^{s_2}\ y$—the bisimulation relation witness is $R = \lambda x\ y.\ \mathsf{True}$. This fact corresponds to the intuition that bisimilarity can only distinguish states through observations along their evolutions, while an unlabeled Markov chain does not produce anything observable. For labeled Markov chains and other systems bisimilarity is a more interesting concept.

The last important notion is that of a *final* F-*coalgebra*: an F-coalgebra for which there exists an unique morphism from any other coalgebra. In our context, the final coalgebra for a BNF F is the destructor $\mathsf{Dtr}_{\mathsf{F}} :: \mathsf{C}_{\mathsf{F}} \Rightarrow \mathsf{C}_{\mathsf{F}}\ \mathsf{F}$ of the codatatype C_{F} induced by F (states of the final coalgebra are of type C_{F}) and the finality is witnessed by the coiterator $\mathsf{unfold}_{\mathsf{F}}$ mapping a coalgebra $s :: \sigma \Rightarrow \sigma\ \mathsf{F}$ to the (unique) function $\mathsf{unfold}_{\mathsf{F}}\ s :: \sigma \Rightarrow \mathsf{C}_{\mathsf{F}}$ satisfying the characteristic equation of a coalgebra morphism: $\mathsf{Dtr}_{\mathsf{F}} \circ \mathsf{unfold}_{\mathsf{F}}\ s = \mathsf{map}_{\mathsf{F}}\ (\mathsf{unfold}_{\mathsf{F}}\ s) \circ s$. The similarity of the coinduction rule for C_{F} to the definition of bisimilarity is not a coincidence: codatatypes are quotients modulo bisimilarity.

Name	BNF	Induced Codatatype
Markov chain	σ pmf	MC
Labeled Markov chain	$\alpha \times \sigma$ pmf	α LMC
Labeled Markov decision process	$\alpha \times \sigma$ pmf set_1^κ	α LMDP^κ
Deterministic automaton	$\alpha \Rightarrow \sigma$ option	α DLTS
Non-deterministic automaton*	$(\alpha \times \sigma)$ set^κ	α LTS^κ
Reactive system	$\alpha \Rightarrow \sigma$ pmf option	α React
Generative system	$(\alpha \times \sigma)$ pmf option	α Gen
Stratified system	σ pmf $+ (\alpha \times \sigma)$ option	α Str
Alternating system	σ pmf $+ (\alpha \times \sigma)$ set^κ	α Alt^κ
Simple Segala system	$(\alpha \times \sigma$ pmf$)$ set^κ	α SSeg^κ
Segala system	$(\alpha \times \sigma)$ pmf set^κ	α Seg^κ
Bundle system	$(\alpha \times \sigma)$ set^κ pmf	α Bun^κ
Pnueli-Zuck system	$(\alpha \times \sigma)$ set^{κ_1} pmf set^{κ_2}	α $\mathsf{PZ}^{\kappa_1,\kappa_2}$
Most general system	$(\alpha \times \sigma + \sigma)$ set^{κ_1} pmf set^{κ_2}	α $\mathsf{MG}^{\kappa_1,\kappa_2}$

* The type $(\alpha \times \sigma)$ set^κ is isomorphic to the more standard $\alpha \Rightarrow \sigma$ set^κ for α set $\leq \kappa$.

Table 1: List of formalized probabilistic systems

Modeled Systems Table 1 lists the systems we consider and their BNFs. (The standard type **datatype** α option = None | Some α is another BNF.) This list contains all the systems from the original probabilistic hierarchy [21], except for Vardi systems, which must be treated separately (§6.4). Moreover, popular systems—labeled Markov chains and Markov decision processes—are new additions. The third column assigns a name to the induced codatatype (e.g., for labeled Markov decision processes, we write **codatatype** α $\mathsf{LMDP}^\kappa = \mathsf{Ctr}_{\mathsf{LMDP}}$ $(\alpha \times \alpha$ LMDP^κ pmf $\mathsf{set}_1^\kappa)$ in Isabelle).

6 The Formalized Hierarchy

How can one compare the expressiveness of the different probabilistic system types? A natural criterion [21] is to exhibit a mapping between the types of systems that preserves and reflects bisimilarity as a witness for an increase in expressiveness along the mapping. Figure 3 shows our formalized hierarchy where arrows represent such mappings. New systems, not analyzed by Sokolova [21], are highlighted with a gray background. Some arrows are annotated with necessary conditions on the bounds of the involved bounded set types. We refer to our formalization [14] for the definitions of all mappings.

Below, we first sketch our proof of the preservation and reflection of bisimilarity abstractly for any mapping. Then we present our formal Isabelle proof for one particular pair of system types and compare our formalized hierarchy with the original [21].

6.1 The Abstract Proof

Formally, for two types of systems given as BNFs F and G, we consider G to be at least as expressive as F, if there is a mapping G_of_F :: σ F $\Rightarrow \sigma$ G that preserves and reflects

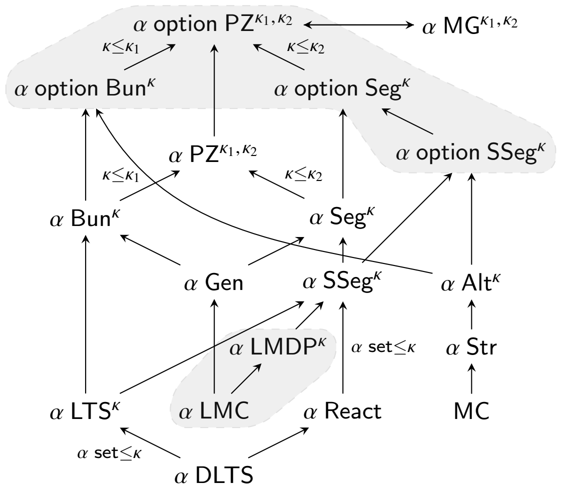


Fig. 3: Probabilistic hierarchy

bisimilarity, i.e., satisfies

$$x \;{}^{s_1}\!\!\sim^{s_2}_{\mathsf{F}}\; y \longleftrightarrow x \;{}^{(\mathsf{G\_of\_F} \circ s_1)}\!\!\sim^{(\mathsf{G\_of\_F} \circ s_2)}_{\mathsf{G}}\; y \tag{3}$$

for all F-coalgebras $s_1, s_2 :: \sigma \Rightarrow \sigma$ F and states $x, y :: \sigma$. Note that by composing the F-coalgebras with G_of_F we obtain G-coalgebras: G_of_F $\circ\, s_1$, G_of_F $\circ\, s_2 :: \sigma \Rightarrow \sigma$ G.

For any mapping G_of_F :: σ F $\Rightarrow \sigma$ G, we prove equation 3 in four steps starting from the right-hand side:

$$\begin{aligned} x \;{}^{(\mathsf{G\_of\_F} \circ s_1)}\!\!\sim^{(\mathsf{G\_of\_F} \circ s_2)}_{\mathsf{G}}\; y &\xleftrightarrow{1} \mathsf{unfold}_{\mathsf{G}}\ (\mathsf{G\_of\_F} \circ s_1)\ x = \mathsf{unfold}_{\mathsf{G}}\ (\mathsf{G\_of\_F} \circ s_2)\ y \\ &\xleftrightarrow{2} \overline{\mathsf{G\_of\_F}}\ (\mathsf{unfold}_{\mathsf{F}}\ s_1\ x) = \overline{\mathsf{G\_of\_F}}\ (\mathsf{unfold}_{\mathsf{F}}\ s_2\ y) \\ &\xleftrightarrow{3} \mathsf{unfold}_{\mathsf{F}}\ s_1\ x = \mathsf{unfold}_{\mathsf{F}}\ s_2\ y \\ &\xleftrightarrow{1} x \;{}^{s_1}\!\!\sim^{s_2}_{\mathsf{F}}\; y \end{aligned}$$

where $\overline{\mathsf{G\_of\_F}} :: \mathsf{C_F} \Rightarrow \mathsf{C_G}$ abbreviates $\mathsf{unfold_G}$ (G_of_F $\circ$ $\mathsf{Dtr_F}$). The first and the last step (labeled with a 1) are both instances of the general fact that for any BNF F, bisimilarity is equivalent to equality on the induced codatatype $\mathsf{C_F}$. Formally, $x \;{}^{s_1}\!\!\sim^{s_2}_{\mathsf{F}}\; y \longleftrightarrow$ $\mathsf{unfold_F}\ s_1\ x = \mathsf{unfold_F}\ s_2\ y$.

In step 2 we perform equational reasoning. The diagram in Figure 4 illustrates the situation. Essentially it shows three commutative diagrams for the characteristic property of the coiterators $\mathsf{unfold_F}$ and $\mathsf{unfold_G}$: one for the F-coalgebra s in the lower left rectangle; one for the G-coalgebra G_of_F $\circ\, s$ using the outermost arrows; and one for the G-coalgebra G_of_F $\circ\, \mathsf{Dtr_F}$ in the upper rectangle.

To make the overall diagram commute, the mapping G_of_F has to be a natural transformation, i.e., commute with the map functions for F and G (lower right rectangle). Once this is ensured we can deduce $\mathsf{unfold_G}$ (G_of_F $\circ\, s$) $= \overline{\mathsf{G\_of\_F}} \circ \mathsf{unfold_F}\ s$ (leftmost “triangle”) and use this equation as a rewrite rule.

$$\begin{tikzcd}
\mathsf{C_G} \arrow[rr, "\mathsf{Dtr_G}"] & & \mathsf{C_G\ G} \\
\mathsf{C_F} \arrow[u, "\overline{\mathsf{G\_of\_F}}"'] \arrow[r, "\mathsf{Dtr_F}"] & \mathsf{C_F\ F} \arrow[r, "\mathsf{G\_of\_F}"] & \mathsf{C_F\ G} \arrow[u, "\mathsf{map_G}\ \overline{\mathsf{G\_of\_F}}"] \\
\alpha \arrow[u, "\mathsf{unfold_F}\ s"'] \arrow[uu, bend left, "\mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s)"] \arrow[r, "s"'] & \alpha\ \mathsf{F} \arrow[u, "\mathsf{map_F}\ (\mathsf{unfold_F}\ s)"] \arrow[r, "\mathsf{G\_of\_F}"'] & \alpha\ \mathsf{G} \arrow[u, "\mathsf{map_G}\ (\mathsf{unfold_F}\ s)"] \arrow[uu, bend right, "\mathsf{map_G}\ (\mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s))"']
\end{tikzcd}$$

Fig. 4: Bisimilarity preservation and reflection via codatatypes

Step 3 holds universally iff $\overline{\mathsf{G\_of\_F}}$ is injective (note that $\mathsf{unfold_F}$ is surjective, since, e.g., $\mathsf{unfold_F}\ \mathsf{Dtr_F} = \mathsf{id}$). In principle, injectivity of $\overline{\mathsf{G\_of\_F}}$ can be further reduced to injectivity of $\mathsf{G\_of\_F}$, which yields the nice abstract characterization from the original hierarchy [2, 21]: if $\mathsf{G\_of\_F}$ is an injective natural transformation then it preserves and reflects bisimilarity. Instead of formalizing the reduction of injectivity of $\overline{\mathsf{G\_of\_F}}$ to the injectivity of $\mathsf{G\_of\_F}$ (which must be done for all concrete instances of $\mathsf{G\_of\_F}$), we found it easier to prove the injectivity of $\overline{\mathsf{G\_of\_F}}$ directly by coinduction. Likewise, instead of chasing the above commutative diagram, we also prove directly the equation $\mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s) = \overline{\mathsf{G\_of\_F}} \circ \mathsf{unfold_F}\ s$ by coinduction.

6.2 A Concrete Example

We consider a particular instantiation for the BNFs F and G: simple Segala systems ($\mathsf{F} = (\alpha \times \sigma\ \mathsf{pmf})\ \mathsf{set}^\kappa$) and Segala systems ($\mathsf{G} = (\alpha \times \sigma)\ \mathsf{pmf}\ \mathsf{set}^\kappa$) and define the mapping $\mathsf{G\_of\_F}\ sseg = \mathsf{map_{set}}\ (\lambda(a, p).\ \mathsf{map_{pmf}}\ (\lambda s.\ (a, s))\ p)\ sseg$.

Next, we formally prove the properties of $\overline{\mathsf{G\_of\_F}}$ outlined in the previous section by straightforward coinductions. We start with the detailed manual proof of the commutation property (leftmost "triangle" in Figure 4).

The proof shown in Figure 5 gives a flavor of the proof obligations that arise with coinduction. The *coinduction* method instantiates the free variable R from the coinduction rule for $\alpha\ \mathsf{Seg}^\kappa$ with the canonical bisimulation witness $\lambda seg\ seg'.\ \exists x.\ seg = \mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s)\ x\ \wedge\ seg' = (\overline{\mathsf{G\_of\_F}} \circ \mathsf{unfold_F}\ s)\ x$ and performs some minimal postprocessing [4]. We are left to prove that $\mathsf{Dtr_G}\ (\mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s)\ x)$ and $\mathsf{Dtr_G}\ (\mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s)\ x)$ are related by the bisimulation witness lifted to the $\alpha\ \mathsf{Seg}^\kappa$-inducing BNF via its relator. This subgoal is easy to discharge by unfolding and resolution. All the used theorems with a dot in their name are generated by the **primrec** and **bnf** commands. The theorem *unfold*$_\mathsf{F}$*.simps* is the characteristic property of the coiterator; theorems *rel-map* (two theorems) and *rel-refl* follow from the BNF properties and are given below for $\alpha\ \mathsf{pmf}$:

$$\mathsf{rel_{pmf}}\ R\ (\mathsf{map_{pmf}}\ f\ p)\ q = \mathsf{rel_{pmf}}\ (\lambda x\ y.\ R\ (f\ x)\ y)\ p\ q$$
$$\mathsf{rel_{pmf}}\ R\ p\ (\mathsf{map_{pmf}}\ g\ q) = \mathsf{rel_{pmf}}\ (\lambda x\ y.\ R\ x\ (g\ y))\ p\ q \qquad (\forall x.\ R\ x\ x) \longrightarrow \mathsf{rel_{pmf}}\ R\ p\ p$$

lemma $\mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s) = \overline{\mathsf{G\_of\_F}} \circ \mathsf{unfold_F}\ s$
proof (*rule ext*)
  fix x
  show $\mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s)\ x = (\overline{\mathsf{G\_of\_F}} \circ \mathsf{unfold_F}\ s)\ x$
  proof (*coinduction arbitrary: x*)
    fix x
    show $\mathsf{rel_{set}}\ (\mathsf{rel_{pmf}}\ (\mathsf{rel_{prod}}\ (=)\ (\lambda seg\ seg'.\ \exists x.\ \ seg = \mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s)\ x\ \wedge$
$seg' = (\overline{\mathsf{G\_of\_F}} \circ \mathsf{unfold_F}\ s)\ x)))$
$(\mathsf{Dtr_G}\ (\mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s)\ x))\ (\mathsf{Dtr_G}\ ((\overline{\mathsf{G\_of\_F}} \circ \mathsf{unfold_F}\ s)\ x))$
    unfolding *unfold$_G$.simps unfold$_F$.simps bset.rel-map pmf.rel-map*
      o-def split-beta map-prod-def rel-prod-apply id-apply fst-conv snd-conv
    by (*rule bset.rel-refl pmf.rel-refl conjI exI refl*)+
  qed
qed

Fig. 5: Isar proof of the commutation property from simple Segala to Segala systems

The proof can be automated by registering the appropriate rules as simplification and introduction rules. Furthermore, it can be seen as a proof template: we have to perform the same reasoning for all concrete mappings that we consider and the only part that is changing is the relator. Fortunately, the Eisbach proof method language [19] helps us to avoid repeating the proof by creating a dedicated proof method, where we replace the manual **unfolding** and *rule* steps by *fastforce*. The proof then collapses to a one-liner.

method-definition *commute-prover* =
  rule ext,
  match conclusion in $u_1\ s_1\ x = (f \circ u_2\ s_2)\ x$ for $f\ u_1\ u_2\ s_1\ s_2\ x \Rightarrow$
    (*coinduction arbitrary: x*, *fastforce*)
lemma $\mathsf{unfold_G}\ (\mathsf{G\_of\_F} \circ s) = \overline{\mathsf{G\_of\_F}} \circ \mathsf{unfold_F}\ s$ **by** *commute-prover*

We treat the injectivity of $\overline{\mathsf{G\_of\_F}}$ and the fact that bisimilarity coincides with equality on codatatypes for F and G in a similar fashion. As before, we omit some essential simplification and introduction rules given as arguments to *fastforce* that make the following degree of automation possible.

method-definition *inj-prover* =
  rule injI,
  match conclusion in $x = y$ for $x\ y \Rightarrow$ (*coinduction arbitrary: x y*, *fastforce*)
lemma $\mathsf{inj}\ \overline{\mathsf{G\_of\_F}}$ **by** *inj-prover*

method-definition $\sim$*-alt-prover* =
  intro iffI, *elim exE conjE*,
  match conclusion in $u_1\ s_1\ x = u_2\ s_2\ y$ for $u_1\ u_2\ s_1\ s_2\ x\ y \Rightarrow$
    (*coinduction arbitrary: x y*, *fastforce*), *fastforce*
lemma $x\ {}^{s_1}\!\!\sim_{\mathsf{F}}^{s_2}\ y \longleftrightarrow \mathsf{unfold_F}\ s_1\ x = \mathsf{unfold_F}\ s_2\ y$ **by** $\sim$*-alt-prover*
lemma $x\ {}^{s_1}\!\!\sim_{\mathsf{G}}^{s_2}\ y \longleftrightarrow \mathsf{unfold_G}\ s_1\ x = \mathsf{unfold_G}\ s_2\ y$ **by** $\sim$*-alt-prover*

Overall, for each of the 14 considered probabilistic system types we prove the alternative bisimilarity characterization **by** $\sim$*-alt-prover* and for each of the 22 mappings (there are 25 arrows in Figure 3, but e.g., the mapping from α option SSeg^{κ} to α option Seg^{κ} is the same as the one from α SSeg^{κ} to α Seg^{κ}) we prove two statements by a one-liner with one of our dedicated methods: *commute-prover* and *inj-prover*. Finally, we state the 25 bisimilarity preservation and reflection properties (equation 3) and prove all of them by equational reasoning (i.e., one line of unfolding). The whole hierarchy is formalized in 450 lines (including the codatatype declarations).

6.3 Comparison to the Original Hierarchy

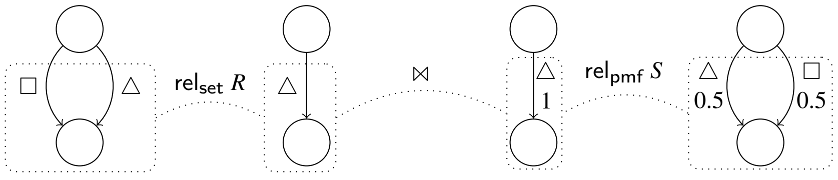
Our formalized hierarchy differs structurally from the original hierarchy [21] in three aspects. First, ours omits the Vardi systems (also known as concurrent Markov chains) for reasons we outline in a separate section (§6.4). Conversely, we have added two popular types of systems, namely labeled Markov chains and Markov decision processes. Furthermore, we observe that the Most General systems α $\mathsf{MG}^{\kappa_1,\kappa_2}$, specifically introduced in the original hierarchy in order to have a top element, are isomorphic to Pnuelli-Zuck systems extended with a single additional label (which we model by using α option instead of just α, i.e., α option $\mathsf{PZ}^{\kappa_1,\kappa_2}$). In other words, no new structurally different probabilistic system is needed to get a top element if one allows additional labels. Following up on this idea, we investigated adding new labels to various other systems in the hierarchy. As a result, Alternating systems α Alt^{κ} are placed below label-extended simple Segala systems α option SSeg^{κ} and Bundle systems α option Bun^{κ}, instead of just below the top element α $\mathsf{MG}^{\kappa_1,\kappa_2}$ as in the original hierarchy.

Our usage of codatatypes (final coalgebras) caters for highly automatable proofs. However, the resulting conciseness comes at a price: final coalgebras need to exist. Concretely, this means that all our systems must be BNFs, in particular bounded and weak pullback preserving. In contrast, the original hierarchy does not require a boundedness assumption (basically allowing to use α set instead of α set^{κ}) and requires for each mapping only the system functor of the mapping's domain to preserve weak pullbacks. While we acknowledge the latter as a limitation of our approach, we point out that the boundedness assumption is not a restriction in the setting of the hierarchy, since the mappings are polymorphic in the type κ used as the bound. That is, for any concrete system with unbounded non-determinism (α set) expressible in HOL we can find an isomorphic bounded one, and the mapping will show how to transform this bounded system into one that is higher in the hierarchy.[1] In contrast, the bounds being part of the types is in some sense more precise—for example, we see that there are two ways of embedding α Alt^{κ} in α $\mathsf{MG}^{\kappa_1,\kappa_2}$ transitively via α option SSeg^{κ} or α option Bun^{κ} and the cardinal bounds give a hint which route was taken.

6.4 Vardi Systems

Vardi systems, also known as concurrent Markov chains [23], blend non-deterministic and probabilistic transitions in a rather symmetric fashion. They are similar to coalge-

[1] Clearly, this discussion is somewhat esoteric, since in practice one barely is interested to look beyond countable sets. Still, we are interested in keeping the results as general as possible.


□
△
$\mathsf{rel}_{\mathsf{set}}\ R$
△
⋈
△
1
$\mathsf{rel}_{\mathsf{pmf}}\ S$
△
0.5
□
0.5
Fig. 6: Intransitivity of Vardi systems

bras of the binary BNF $(\alpha, \sigma)\ \mathsf{Var}_0^\kappa = (\alpha \times \sigma)\ \mathsf{pmf} + (\alpha \times \sigma)\ \mathsf{set}^\kappa$; however there is a twist: Vardi systems identify the singleton bounded set $\{(a, s)\}$ with the singleton discrete distribution $\mathsf{return}_{\mathsf{pmf}}\ (a, s)$. Formally, we define the equivalence relation $\bowtie$ inductively by the following three rules, where Inl and Inr are the sum type embeddings.

$$v \bowtie v \qquad \mathsf{Inl}\ (\mathsf{return}_{\mathsf{pmf}}\ (a, s)) \bowtie \mathsf{Inr}\ \{(a, s)\} \qquad \mathsf{Inr}\ \{(a, s)\} \bowtie \mathsf{Inl}\ (\mathsf{return}_{\mathsf{pmf}}\ (a, s))$$

The type $(\alpha, \sigma)\ \mathsf{Var}^\kappa$ is then defined as a quotient of $(\alpha, \sigma)\ \mathsf{Var}_0^\kappa$ by $\bowtie$. Lifting the functorial structure of $(\alpha, \sigma)\ \mathsf{Var}_0^\kappa$ to the quotient $(\alpha, \sigma)\ \mathsf{Var}^\kappa$ is straightforward and we omit the definitions. However, it turns out that the resulting quotient is not a BNF: its canonical relator $\mathsf{rel}_{\mathsf{Var}}$ does not distribute over relation composition. We only try to convey the intuition behind this fact—a formal proof can be found in our formalization. Figure 6 shows on the left two Vardi automata that use only non-deterministic transitions and are related by $\mathsf{rel}_{\mathsf{set}}\ R$ (lifted to the sum type $(\alpha, \sigma)\ \mathsf{Var}_0^\kappa$ and further to the quotient type $(\alpha, \sigma)\ \mathsf{Var}^\kappa$) where $R\ x\ y \longleftrightarrow y = \triangle$. Similarly, the two automata on the right are related by $\mathsf{rel}_{\mathsf{pmf}}\ S$ where where $S\ x\ y \longleftrightarrow x = \triangle$. The two middle automata are related by $\bowtie$, i.e., they are equal on the quotient type $(\alpha, \sigma)\ \mathsf{Var}^\kappa$. Distributivity of the relator requires the two outermost automata to be related by $\mathsf{rel}_{\mathsf{Var}}$, but this is not the case.

Since $(\alpha, \sigma)\ \mathsf{Var}^\kappa$ is not a BNF, our proof approach is not applicable. Not only that, the above counterexample, found in the course of formalization, is easily transferable into the general coalgebraic setting, allowing us to prove that the functor used in the original hierarchy [21] does not preserve weak pullbacks, and as a consequence bisimilarity of Vardi systems is not an equivalence relation. The weak pullback preservation, however, is a necessary criterion for the original proof method for two mappings from Vardi to Segala and Bundle systems. Those outgoing "arrows" must be purged: there is no such bisimilarity preserving and reflecting mapping.

In contrast to our approach, the original proof still covers the two incoming "arrows" from non-deterministic automata and generative systems to Vardi systems. We have formalized those bisimilarity preserving and reflecting mappings separately, without going through codatatypes. The proofs are significantly longer (overall 145 lines for just two mappings, contrasting 450 lines for 25 mappings in our hierarchy) and less suited for automation, because they require several non-trivial quantifier instantiations. In summary, equational reasoning on codatatypes proved superior whenever applicable.

7 Further Related Work

In the other sections of the paper, we have already referenced existing work we build on, in particular Sokolova's [2,21] and the Isabelle packages and tools [4–6,13,15,19].

Our formalization of pmfs is similar to the work in Coq presented by Audebaud and Paulin-Mohring [1]. They introduce a monadic structure on subprobability measures. They use integrals as representations of measures, while in our case we directly lift measures from Isabelle's measure theory. As their measures are subprobabilities they also provide a fixed-point operator which is not available in general for α pmf. Their formalization is also directed towards program verification; they do not provide a functorial structure (i.e. $\mathsf{map}_{\mathsf{pmf}}$ and $\mathsf{set}_{\mathsf{pmf}}$ in our case) on their type of measures.

Apart from process algebra [16], the relator $\mathsf{rel}_{\mathsf{pmf}}$ is used in probabilistic relational Hoare logic, too [3]. In this context, Zanella [26] proved distributivity with composition in Coq; see §4 for a comparison. Deng [7] collects further results on the relator and its applications. Beyond (strong) bisimilarity, weak bisimilarity compares systems modulo certain irrelevant *invisible* observations. Sokolova [21] recasts weak bisimilarity as bisimilarity of translated systems, which in turn can be hierarchized as presented here.

Mechanizations of category theory abound (see [9] for an overview), and the hierarchy result could probably be formalized with some of them. Yet, we do not formalize the general theory, but its application to concrete instances. Thus, our system types are proper HOL types and can be used directly for modeling concrete systems.

8 Conclusion

We have presented a formalization of the hierarchy of probabilistic system types in Isabelle/HOL. The hierarchy stems from the coalgebraic framework, which presents the various systems in a uniform fashion and caters for simple and concise proofs. We model probabilistic systems as codatatypes, which enables convenient equational reasoning and makes the proofs even more concise. This modeling requires nested corecursion through bounded sets and discrete probability distributions—a perfect match for demonstrating the flexibility of Isabelle's new codatatype facility. Finally, we have learned that weak pullback preservation is an important but subtle property, by uncovering two mistakes in informal proofs.

Acknowledgment. We thank Tobias Nipkow for supporting this collaboration and Ana Sokolova for confirming our findings regarding Vardi systems. Jasmin Blanchette, Ondřej Kunčar and anonymous reviewers helped to improve the presentation through numerous comments and offered stylistic advice. Hölzl is supported by the DFG project Verification of Probabilistic Models in Interactive Theorem Provers (grant Ni 491/15-1). Traytel is supported by the DFG program Program and Model Analysis (doctorate program 1480). The authors are listed alphabetically.

References

1. Audebaud, P., Paulin-Mohring, C.: Proofs of randomized algorithms in Coq. Sci. of Comput. Program. 74(8), 568–589 (2009)
2. Bartels, F., Sokolova, A., de Vink, E.P.: A hierarchy of probabilistic system types. Theor. Comput. Sci. 327(1-2), 3–22 (2004)
3. Barthe, G., Fournet, C., Grégoire, B., Strub, P.Y., Swamy, N., Zanella Béguelin, S.: Probabilistic relational verification for cryptographic implementations. In: Jagannathan, S., Sewell, P. (eds.) POPL 2014. pp. 193–205. ACM (2014)

4. Blanchette, J.C., Hölzl, J., Lochbihler, A., Panny, L., Popescu, A., Traytel, D.: Truly modular (co)datatypes for Isabelle/HOL. In: Klein, G., Gamboa, R. (eds.) ITP 2014. LNCS, vol. 8558, pp. 93–110. Springer (2014)
5. Blanchette, J.C., Popescu, A., Traytel, D.: Cardinals in Isabelle/HOL. In: Klein, G., Gamboa, R. (eds.) ITP 2014. LNCS, vol. 8558, pp. 111–127. Springer (2014)
6. Blanchette, J.C., Popescu, A., Traytel, D.: Witnessing (co)datatypes. In: Vitek, J. (ed.) ESOP 2015. LNCS, vol. 9032, pp. 359–382. Springer (2015)
7. Deng, Y.: Semantics of Probabilistic Processes. Springer (2014)
8. Eberl, M., Hölzl, J., Nipkow, T.: A verified compiler for probability density functions. In: Vitek, J. (ed.) ESOP 2015. LNCS, vol. 9032, pp. 80–104. Springer (2015)
9. Gross, J., Chlipala, A., Spivak, D.I.: Experience implementing a performant category-theory library in Coq. In: Klein, G., Gamboa, R. (eds.) ITP 2014. LNCS, vol. 8558, pp. 275–291. Springer (2014)
10. Gunter, E.L.: Why we can't have SML-style `datatype` declarations in HOL. In: TPHOLs '92. IFIP Transactions, vol. A-20, pp. 561–568. North-Holland/Elsevier (1993)
11. Harrison, J.: A HOL theory of Euclidean space. In: Hurd, J., Melham, T.F. (eds.) TPHOLs 2005. LNCS, vol. 3603, pp. 114–129. Springer (2005)
12. Hermida, C., Jacobs, B.: Structural induction and coinduction in a fibrational setting. Inf. Comput. 145(2), 107–152 (1998)
13. Hölzl, J.: Construction and Stochastic Applications of Measure Spaces in Higher-Order Logic. Ph.D. thesis, Institut für Informatik, Technische Universität München (2013)
14. Hölzl, J., Lochbihler, A., Traytel, D.: A zoo of probabilistic systems. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. `http://afp.sourceforge.net/entries/Probabilistic_System_Zoo.shtml` (2015)
15. Huffman, B., Kunčar, O.: Lifting and Transfer: A modular design for quotients in Isabelle/HOL. In: Gonthier, G., Norrish, M. (eds.) CPP 2013. LNCS, vol. 8307, pp. 131–146. Springer (2013)
16. Jonsson, B., Larsen, K.G., Yi, W.: Probabilistic extensions of process algebras. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebras, chap. 11, pp. 685–710. Elsevier (2001)
17. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Inf. and Comp. 94(1), 1–28 (1991)
18. Lochbihler, A.: Measure definition on streams. Archived at `https://lists.cam.ac.uk/pipermail/cl-isabelle-users/2015-February/msg00112.html` (24 Feb 2015)
19. Matichuk, D., Wenzel, M., Murray, T.C.: An Isabelle proof method language. In: Klein, G., Gamboa, R. (eds.) ITP 2014. LNCS, vol. 8558, pp. 390–405. Springer (2014)
20. Rutten, J.J.M.M.: Universal coalgebra: A theory of systems. Theor. Comput. Sci. 249, 3–80 (2000)
21. Sokolova, A.: Coalgebraic Analysis of Probabilistic Systems. Ph.D. thesis, Technische Universiteit Eindhoven (2005)
22. Traytel, D., Popescu, A., Blanchette, J.C.: Foundational, compositional (co)datatypes for higher-order logic—Category theory applied to theorem proving. In: LICS 2012. pp. 596–605. IEEE (2012)
23. Vardi, M.Y.: Automatic verification of probabilistic concurrent finite-state programs. In: FOCS 1985. pp. 327–338. IEEE (1985)
24. de Vink, E.P., Rutten, J.J.: Bisimulation for probabilistic transition systems: a coalgebraic approach. Theor. Comput. Sci. 221(1-2), 271–293 (1999)
25. Weber, T.: Introducing a BNF for sets of bounded cardinality. Archived at `https://lists.cam.ac.uk/pipermail/cl-isabelle-users/2015-March/msg00116.html` (14 Mar 2015)
26. Zanella Béguelin, S.: Formal Certification of Game-Based Cryptographic Proofs. Ph.D. thesis, École Nationale Supérieure des Mines de Paris (2010)