Logical Methods in Computer Science
Vol. 14(1:19)2018, pp. 1–32
https://lmcs.episciences.org/

Submitted Dec. 16, 2016
Published Feb. 28, 2018

# A COALGEBRAIC TREATMENT OF CONDITIONAL TRANSITION SYSTEMS WITH UPGRADES

HARSH BEOHAR [a], BARBARA KÖNIG [a,*], SEBASTIAN KÜPPER [b], ALEXANDRA SILVA [c,⋆], AND THORSTEN WISSMANN [d,†]

[a] Universität Duisburg-Essen, Germany
*e-mail address*: {harsh.beohar,barbara_koenig}@uni-due.de

[b] FernUniversität Hagen, Germany
*e-mail address*: sebastian.kuepper@fernuni-hagen.de

[c] University College London, United Kingdom
*e-mail address*: alexandra.silva@ucl.ac.uk

[d] Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
*e-mail address*: thorsten.wissmann@fau.de

*Dedicated to Jiří Adámek on the occasion of his 70th birthday*

Abstract. We consider conditional transition systems, that model software product lines with upgrades, in a coalgebraic setting. By using Birkhoff's duality for distributive lattices, we derive two equivalent Kleisli categories in which these coalgebras live: Kleisli categories based on the reader and on the so-called lattice monad over Poset. We study two different functors describing the branching type of the coalgebra and investigate the resulting behavioural equivalence. Furthermore we show how an existing algorithm for coalgebra minimisation can be instantiated to derive behavioural equivalences in this setting.

## 1. Introduction

Jiří Adámek has made many important contributions to category theory to the theory of coalgebras. The final (or terminal) chain to construct the final coalgebra [AK95] will play a key role in this paper. In addition Jiří Adámek wrote, jointly with Horst Herrlich and George E. Strecker, the well-known textbook "Abstract and Concrete Categories – The Joy of Cats" [AHS90], which has served as an invaluable guide to us when learning and looking up results on category theory, also for this paper.

It is a continuation of the work that two of the co-authors did jointly with Jiří Adámek [ABH+12]. In that paper we studied generic versions of minimisation and determinisation

*Key words and phrases:* conditional transition systems, coalgebras, behavioural equivalence, Kleisli categories.

* Supported by the DFG-funded project BEMEGA (KO 2185/7-1).
⋆ Supported by the ERC starting grant ProFoundNet (679127).
† Supported by the DFG-funded project COAX (MI 717/5-1).


DOI:10.23638/LMCS-14(1:19)2018

algorithms in the context of coalgebras, especially in Kleisli categories. Here we are studying a novel type of transition system, called conditional transition systems, and show how they fit into this framework.

This example is interesting for several reasons: first, it gives a non-trivial case study in coalgebra which demonstrates the generality of the approach. Second, it studies coalgebras in the category of partially ordered sets, respectively in Kleisli categories over this base category. We use the Birkhoff duality for distributive lattices to show the equivalence of two Kleisli categories over two monads: the reader monad and the so-called lattice monad. This result can be of interest, independently of the coalgebraic theory. Third, we introduce a notion of upgrade into coalgebraic modelling.

The theory of coalgebras [Rut00] allows uniform modelling and reasoning for a variety of state-based systems. For instance, (non)deterministic finite automata and weighted automata are classical examples often studied in this context (see [Rut00] for more examples). Furthermore, coalgebraic modelling comes with the benefit of offering generic algorithms, capturing the core of algorithms that are similar across different types of automata. In particular, the final-chain based algorithm [AK95] computes quotients on automata up to a chosen notion of behavioural equivalence (such as strong bisimilarity or trace equivalence).

A conditional transition system (CTS) [ABH$^+$12, BKKS17] is an extension of a labelled transition system that is well suited to model software product lines [CN01], an emergent topic of research in the field of software engineering. In contrast to the commonly used featured transition systems [CCS$^+$13], CTSs are not primarily concerned with the individual features of a software product, but mainly with the individual versions that may arise from the given feature combinations.

In CTSs [BKKS17] transitions are labelled with the elements of a partially ordered set of conditions $(\Phi, \leq_\Phi)$, which can be viewed as software products in the terminology of software product lines. This gives us a compact representation which merges the transition systems for many different versions into one single structure. A transition labelled $\varphi \in \Phi$ can only be taken in version $\varphi$. Furthermore, with $\varphi' \leq_\Phi \varphi$ we denote that – during execution – version $\varphi$ can be upgraded to $\varphi'$.

Intuitively CTSs evolve in two steps: first, a condition $\varphi \in \Phi$ is chosen at a given state; second, a transition is fired which is guarded by the chosen condition. Over the course of the run of a CTS, it can perform an operation called *upgrade* in which the system changes from a greater condition $\varphi$ to a smaller condition $\varphi' \leq_\Phi \varphi$. This in turn activates additional transitions that may be taken in future steps. Originally, CTSs in [ABH$^+$12] were defined without upgrades, i.e., $\leq_\Phi$ was fixed to be equality.

CTS have 'monotonous' upgrading in the sense that one can only go down on the hierarchy of conditions, but not up. As a consequence, CTSs have a special notion of bisimulation consisting of a family of traditional bisimulations $\sim_\varphi$ (one for each condition $\varphi \in \Phi$) such that $\sim_\varphi \subseteq \sim_{\varphi'}$, whenever $\varphi' \leq_\Phi \varphi$. Roughly, two states are behaviourally equivalent under a condition $\varphi$ if and only if they are bisimilar (in the traditional sense) for every upgrade $\varphi' \leq_\Phi \varphi$. An interesting fact about a CTS is that there exists an equivalent model, called *lattice transition system* (LaTS), which allows for a more compact representation of a CTS using the lattice of downward closed subsets of $\Phi$ (see [BKKS17] for more details). In essence, this can be viewed as a lifting of the well-known Birkhoff's representation theorem to the case of transition systems.

This paper aims at characterising CTS and LaTS coalgebraically. To this end, we define two monads, the reader monad and the lattice monad, which allow for modelling CTS and

LaTS respectively – provided a matching functor is chosen – in their corresponding Kleisli categories. We will show that these two categories are equivalent.

Our next aim is to characterise conditional bisimilarity using the notion of behavioural equivalence, a concept stemming from the theory of coalgebras. Roughly, two states of a system (modelled as a coalgebra) are *behaviourally equivalent* if and only if they are mapped to a common point by a coalgebra homomorphism.

In this regard, capturing the right notion of behavioural equivalence (conditional bisimilarity in our case) depends on making the right choice of functor modelling CTSs. By working in a Kleisli category, we are interested in establishing a functor via an extension of a functor on the base category $\mathsf{Poset}$. The usual powerset functor $\mathcal{P}$ proves to be a viable choice for CTSs without any upgrades, but we will provide a counterexample which shows that this functor does not yield conditional bisimulation in the presence of upgrades, no matter how the extension is chosen. However, for an adaptation of the powerset functor, namely $\mathcal{P}(\_ \times \Phi)$, behavioural equivalence indeed captures conditional bisimilarity in the presence of upgrades. Our approach is not restricted to the treatment of those two specific functors: we introduce so-called *version filters* that add conditions/versions to any $\mathsf{Poset}$ functor and also develop an abstract machinery to capture conditional bisimilarity coalgebraically.

To conclude, we show that the minimisation algorithm based on the final chain construction plus factorisation structures [ABH$^+$12] is applicable to the category under investigation and specify how it can be applied to CTSs. CTSs without upgrades have already been considered in [ABH$^+$12], but applicability to CTSs with upgrades is novel.

This paper is structured as follows: in Section 2 (*Preliminaries*), we will define coalgebras with their notion of behavioural equivalence. In the coalgebraic treatment of conditional transition systems we view the currently chosen software product (also called condition) $\varphi \in \Phi$ as a form of side effect and we will work with Kleisli categories for the reader monad $(\_^{\Phi})$ in order to capture this phenomenon. Hence, we will derive the reader monad on $\mathsf{Poset}$ (working in $\mathsf{Poset}$ is necessary in order to capture upgrades) via the product comonad in $\mathsf{Poset}$. Furthermore we discuss the (known) relationship between distributive laws and extensions of a functor to a Kleisli category.

Then, in Section 3 (*Conditional and Lattice Transition Systems*), we introduce conditional and lattice transition systems and the associated notion of conditional bisimulation from [BKKS17]. The duality between these two variants depends on the Birkhoff duality from lattice theory, which is also reviewed in this section.

While conditional transition systems will be modelled in the Kleisli category for the reader monad, it is not so obvious in which category lattice transition systems should live. In order to solve this question we introduce in Section 4 (*The Lattice Monad*) the lattice monad, which characterises a monotone function $f\colon \Phi \to X$ as a mapping from $X$ into the downsets of $\Phi$ (which form a lattice $\mathbb{L}$). However, simply taking the monad $\mathbb{L}^{(\_)}$ would not be equivalent to $\_^{\Phi}$. Hence we impose suitable restrictions on mappings $\mathbb{L}^X$ and obtain a monad isomorphic to the reader monad. As a result, the two corresponding Kleisli categories are also isomorphic. Not surprisingly, in the case of a finite set $\Phi$ of conditions this isomorphism between the two monads is related to the Birkhoff duality.

In Section 5 (*Modelling Conditional Transition Systems as Coalgebras*) we (first) model conditional transition systems, where the upgrade order is discrete, i.e., the corresponding lattice of downsets is a Boolean algebra. The corresponding coalgebras are Kleisli arrows of the form $X \to\!\!\circ\, (\mathcal{P}X)^A$ in $\mathsf{Kl}(\_^{\Phi})$, where $\mathcal{P}$ is the powerset functor and $A$ is the label

alphabet. (Note that Kleisli arrows are denoted by $-\!\circ\!\!\to$.) In order to be able to define such coalgebras, we have to extend the functor $(\mathcal{P}\_)^A$ (defined on $\mathsf{Poset}$) to $\mathsf{Kl}(\_^\Phi)$ via a distributive law. Furthermore we also consider extensions of the functor $(\mathcal{P}(\_ \times \Phi))^A$, which is required to capture upgrades. Since distributive laws are easier to derive for the comonad $\Phi \times \_$, we consider distributive laws in the general setting of monad-comonad adjunctions. This gives us suitable functor extensions (see Section 5.1).

After suitably extending the functors for both cases (without and with upgrades), we study coalgebraic behavioural equivalences (see Section 5.2). The aim is to eventually show that we capture bisimulation for conditional transition systems in a coalgebraic setting. We go further than that and define conditional bisimulation and congruence for more general behavioural functors. In particular, we introduce version filters that add conditions (from $\Phi$) to any $\mathsf{Poset}$ functor. Then we can prove general results for so-called upgrade-preserving coalgebras that allow us to state the main theorem, namely that we correctly characterize the notion of conditional bisimulation in our abstract setting.

Afterwards, in Section 6 (*Computing Behavioural Equivalence*), we consider an application of this result. In particular, we use a generalized partition refinement algorithm from [ABH+12] to minimise a given coalgebra and to answer questions concerning behavioural equivalence based on this minimisation. This minimisation procedure is based on pseudo-factorisations, i.e., factorisations that are obtained by mapping an arrow into a reflective subcategory, following by factorisation. We show that we have such a reflective subcategory, resulting in suitable pseudo-factorisations and that the algorithm can hence be applied. We work out an example and we compare with the matrix multiplication algorithm in [BKKS17].

Finally, we wrap up the paper and give directions for future work in Section 7 (*Conclusion*).

## 2. Preliminaries

We assume a basic knowledge of category theory. The primary objects of interest in this work are coalgebras, which we use to model conditional transition systems.

**Definition 2.1** (Coalgebra)**.** Let $H\colon \mathcal{C} \to \mathcal{C}$ be an endofunctor on a category $\mathcal{C}$. Then an *$H$-coalgebra* is a pair $(X, \alpha)$, where $X$ is an object of $\mathcal{C}$ and $\alpha\colon X \to HX$ is an arrow in $\mathcal{C}$. An *$H$-coalgebra homomorphism* between two coalgebras $(X, \alpha)$ and $(Y, \beta)$ is an arrow $f\colon X \to Y$ in $\mathcal{C}$ such that $\beta \cdot f = Hf \cdot \alpha$.

$$\begin{array}{ccc} X & \xrightarrow{\alpha} & HX \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle Hf} \\ Y & \xrightarrow{\beta} & HY \end{array}$$

The $H$-coalgebras and their homomorphisms form a category. In the sequel, we drop the prefix '$H$-' whenever it is clear from the context.

In the theory of coalgebras, bisimulation [Par81] is captured in more than one way, namely: coalgebraic bisimulation or via an arrow into any coalgebra (so-called cocongruences). At this stage, we fix the notion of behavioural equivalence in a category $\mathcal{C}$ structured over the category of sets $\mathsf{Set}$ using a concretisation functor $U\colon \mathcal{C} \to \mathsf{Set}$.

**Definition 2.2** (Behavioural Equivalence)**.** Let $F$ be an endofunctor on a concrete category $\mathcal{C}$ with a faithful functor $U\colon \mathcal{C} \to \mathsf{Set}$ to the category of sets $\mathsf{Set}$. Then, two states $x \in UX$ and $x' \in UX$ of a coalgebra $(X, \alpha)$ are *behaviourally equivalent* if there exists a coalgebra homomorphism $f\colon X \to Y$ such that $Uf(x) = Uf(x')$.

**Example 2.3.** In the sequel, we work with the concrete category of partially ordered sets (a.k.a. posets), denoted $\mathsf{Poset}$, as our base category $\mathcal{C}$. Formally, the objects of $\mathsf{Poset}$ are

pairs $(X, \leq_X)$ of a set $X$ and a partial order $\leq_X \subseteq X \times X$; while its arrows are all the order preserving functions between any two posets. If the order relation is just the equality, then we call the poset *discrete*.

**Notation 2.4.** A functor $F\colon \mathcal{C} \to \mathcal{D}$ is *left adjoint* to a functor $U\colon \mathcal{D} \to \mathcal{C}$ (or $U$ is *right adjoint* to $F$), denoted $F \dashv U$, when for any two objects $X$ from $\mathcal{C}$ and $Y$ from $\mathcal{D}$ there is a natural bijection between morphisms

$$\frac{f\colon X \to UY}{g\colon FX \to Y}$$

in the sense that each morphism $f$ (displayed above) uniquely determines a morphism $g$ and conversely. More formally, $F \dashv U$ when there exists a family of isomorphisms

$$\Psi_{X,Y}\colon \mathcal{C}(X, UY) \cong \mathcal{D}(FX, Y)$$

natural in $X$ and $Y$. Lastly, given an adjunction $F \dashv U$, we note that the unit $\rho$ and the counit $\epsilon$ of this adjunction is given by:

$$\rho_X := \Psi^{-1}_{X,FX}(\mathrm{id}_{FX}) \quad \text{and} \quad \epsilon_Y := \Psi_{UY,Y}(\mathrm{id}_{UY})$$

Recall that a monad on $\mathcal{C}$ is a functor $T\colon \mathcal{C} \to \mathcal{C}$ with natural transformations $\eta\colon \mathrm{Id} \to T$ (called *unit*), $\mu\colon TT \to T$ (called *multiplication*) such that $\mu \cdot T\eta = \mathrm{Id} = \mu \cdot \eta T$ and $\mu \cdot T\mu = \mu \cdot \mu T$. Dually, a comonad on $\mathcal{C}$ is a monad on $C^{\mathsf{op}}$, i.e. a functor $T\colon \mathcal{C} \to \mathcal{C}$ with counit $T \to \mathrm{Id}$ and comultiplication $T \to TT$ fulfilling the corresponding laws.

**Proposition 2.5.** *Given a comonad $(L, \vartheta, \delta)$ on a category $\mathcal{C}$ and a functor $T\colon \mathcal{C} \to \mathcal{C}$ such that $L \dashv T$ with unit and counit $\rho$ and $\epsilon$, respectively. Then, this adjunction induces a monad structure on $T$ as follows:*

$$\frac{LX \xrightarrow{\vartheta_X} X}{X \xrightarrow{\eta_X} TX} \qquad \frac{LTTX \xrightarrow{\delta_{TTX}} LLTTX \xrightarrow{L\epsilon_{TX}} LTX \xrightarrow{\epsilon_X} X}{TTX \xrightarrow{\mu_X} TX}$$

For instance, the reader monad is defined in terms of a comonad (see e.g. [PG14, Example 3.10]).

**Definition 2.6** (Reader monad)**.** We have a comonad on $\_ \times \Phi$ with counit $\pi_1\colon X \times \Phi \to X$ and comultiplication $\mathrm{id}_X \times \Delta_\Phi\colon X \times \Phi \to X \times \Phi \times \Phi$ where $\Delta_\Phi$ is the diagonal $\Delta_\Phi\colon \Phi \xrightarrow{\langle \mathrm{id}_\Phi, \mathrm{id}_\Phi \rangle} \Phi \times \Phi$. Using $\_ \times \Phi \dashv \_^\Phi$ with the counit $\mathrm{ev}_X\colon X^\Phi \times \Phi \to X$ on $\mathsf{Poset}$, Proposition 2.5 provides a monad structure $(\_^\Phi, \nu, \zeta)$. Explicitly, we have:

$$X^\Phi = (\mathsf{Poset}(\Phi, X), \leq_{X^\Phi}) \quad (f\colon X \to Y)^\Phi = \mathsf{Poset}(\Phi, f) = (C \mapsto f \cdot C),$$

where $C \leq_{X^\Phi} C'$ if $\forall_{\varphi \in \Phi}\ C(\varphi) \leq_X C'(\varphi)$ and

$$\nu_X(x)(\varphi) = x \quad \text{for } x \in X, \varphi \in \Phi \qquad \zeta_X(D)(\varphi) = D(\varphi)(\varphi) \quad \text{for } D \in X^{\Phi^\Phi}, \varphi \in \Phi.$$

**Notation 2.7.** Given an arrow $f\colon X \times \Phi \to Y$, $\bar{f}\colon X \to Y^\Phi$ denotes its curried version. Furthermore, given an arrow $g\colon X \to Y^\Phi$, $\check{g}\colon X \times \Phi \to Y$ denotes its uncurried version. Lastly, given a monotone map $f\colon X \to Y^\Phi$ then we fix one argument $\varphi \in \Phi$ by writing $f_\varphi\colon X \to Y$ defined as $f_\varphi(x) = f(x)(\varphi)$.

From the seminal work of Moggi [Mog91], it is common to model computations with side-effects by a monad. Generally, such a computation with side-effects in $T$ is treated as an arrow in the Kleisli category of $T$.

**Definition 2.8.** Let $(T, \eta, \mu)$ be a monad on $\mathcal{C}$. Then its *Kleisli category* $\mathsf{Kl}(T)$ has the same objects as $\mathcal{C}$ and the arrows $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$ in $\mathsf{Kl}(T)$ are the arrows $f\colon X \to TY$ in $\mathcal{C}$. The identity on $X$ in $\mathsf{Kl}(T)$ is given by $\eta_X\colon X \mathrel{-\!\!\circ\!\!\to} X$ and the composition of two arrows $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$, $g\colon Y \mathrel{-\!\!\circ\!\!\to} Z$ in $\mathsf{Kl}(T)$ is given by the following composition in $\mathcal{C}$

$$g \circ f := \big( \; X \xrightarrow{f} TY \xrightarrow{Tg} TTZ \xrightarrow{\mu_Z} TZ \; \big).$$

Throughout the paper, we reserve $\circ$ for Kleisli composition, whereas $\cdot$ denotes the composition in the base category $\mathcal{C}$. The base category sits in $\mathsf{Kl}(T)$ witnessed by the functor $I\colon \mathcal{C} \to \mathsf{Kl}(T)$ defined as follows: $I(X) = X$, for each object $X$; $I(f) = \eta_Y \cdot f$, for each arrow $f\colon X \to Y$ in $\mathcal{C}$. If $\eta$ has monic components, then this functor is faithful, i.e., $\mathcal{C}$ is a subcategory of $\mathsf{Kl}(T)$. A Kleisli arrow $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$ is called *pure*, if $f\colon X \to TY$ factors through $\eta_Y\colon Y \to TY$ in $\mathcal{C}$, i.e., if there is some arrow $f'\colon X \to Y$ with $If' = f$. Inuitively speaking, pure arrows have no side-effects. The subcategory of pure Kleisli arrows is precisely $\mathcal{C}$. The Kleisli composition of $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$ with pure maps boils down to the composition in $\mathcal{C}$:

$$\begin{aligned} f \circ Ip &= f \cdot p && \text{for } p\colon P \to X \\ Ip \circ f &= Tp \cdot f && \text{for } p\colon X \to P. \end{aligned}$$

When considering coalgebras on a Kleisli category, one can distinguish the visible effects of transitions in a system from the side-effects. For instance, when checking the language equivalence of two states in a nondeterministic automaton, one only cares about the final states and the consumed input word, but not about the non-deterministic branching.

While determining the behavioural equivalence of interest, the intended observable effects of a transition are encoded in an endofunctor $F$ on the Kleisli category; whereas, the side effects are encoded via a monad $T$. This is motivated by the previous works in [HJS07, PT99], where behavioural equivalence in Kleisli categories were used to characterise (trace) language equivalence (rather than bisimulation).

Notwithstanding, the endofunctor $F$ and the monad $T$ of interest are often defined on the base category. Thus, one needs a mechanism to *extend* the given functor $F$ as an endofunctor $\hat{F}$ on the Kleisli category $\mathsf{Kl}(T)$.

**Definition 2.9.** An *extension* of a functor $F\colon \mathcal{C} \to \mathcal{C}$ to $\mathsf{Kl}(T)$ is a functor $\hat{F}\colon \mathsf{Kl}(T) \to \mathsf{Kl}(T)$ such that $IF = \hat{F}I$. A distributive law $\lambda\colon FT \to TF$ is a natural transformation $FT \to TF$ that preserves the monad structure of $T$ in the obvious way.

$$\begin{array}{ccc} \mathsf{Kl}(T) & \xrightarrow{\hat{F}} & \mathsf{Kl}(T) \\ {\scriptstyle I}\uparrow & & \uparrow{\scriptstyle I} \\ \mathcal{C} & \xrightarrow{F} & \mathcal{C} \end{array}$$

We end this section by recalling a standard result on distributive laws from [HJS07, Mul94].

**Theorem 2.10.** *For a functor $F\colon \mathcal{C} \to \mathcal{C}$ there is a one-to-one correspondence between:*
(1) *Extensions $\hat{F}\colon \mathsf{Kl}(T) \to \mathsf{Kl}(T)$ of $F$.*
(2) *Distributive law $\lambda\colon FT \to TF$ for $F$.*

*Given an extension, the corresponding distributive law is $\hat{F}(\mathrm{id}_X\colon TX \mathrel{-\!\!\circ\!\!\to} X)\colon FTX \to TFX$ and conversely a a distributive law defines an extension by*

$$\hat{F}(f\colon X \mathrel{-\!\!\circ\!\!\to} Y) = \big(FX \xrightarrow{Ff} FTY \xrightarrow{\lambda_Y} TFY\big).$$

**Proposition 2.11.** *Given a monad $T$ whose unit $\eta$ has monic components, then a functor $\hat{F}\colon \mathsf{Kl}(T) \to \mathsf{Kl}(T)$ is an extension of some functor $F : \mathcal{C} \to \mathcal{C}$ iff $\hat{F}$ preserves pure morphisms.*

*Proof.*

($\Rightarrow$) The square of $\hat{F}$ being an extension of $F$ directly says that $\hat{F}$ maps any pure morphism $If$ to the pure morphism $IFf$.

($\Leftarrow$) On objects we put $FX := \hat{F}X$. Let $f\colon X \to Y$. Then $\hat{F}$ maps $If\colon X \mathrel{-\!\!\circ\!\!\to} Y$ to the pure $\hat{F}If\colon \hat{F}X \mathrel{-\!\!\circ\!\!\to} \hat{F}Y$, so there is some $g\colon FX \to FY$ with $Ig = \hat{F}If$. Since $\eta$ has monic components, $I$ is faithful and there is a unique such $g$. Hence we can put $Ff := g$ and thus have $IFf = Ig = \hat{F}If$. Since $\eta_X$ is monic, $\mathrm{id}_{FX}$ is the only morphism $g\colon X \to X$ with $Ig = I\mathrm{id}_X = \eta_X$. Using the faithfulness of $I$, $F$ preserves composition. □

## 3. Conditional and lattice transition systems

Here we recall the definitions of a CTS, a LaTS, and conditional bisimilarity from [BKKS17].

**Definition 3.1.** A *conditional transition system* (CTS) is a tuple $(X, A, \Phi, f)$ consisting of a set of states $X$, a set of actions $A$, a finite set of conditions $\Phi$, and a transition function $f\colon X \times A \to (\mathcal{P}X, \supseteq)^{(\Phi, \leq_\Phi)}$ that maps every pair $(x, a) \in X \times A$ to a monotone function of type $(\Phi, \leq_\Phi) \to (\mathcal{P}X, \supseteq)$. We write $x \xrightarrow{a,\varphi} x'$, whenever $x' \in f(x, a)(\varphi)$. In case $|A| = 1$, we omit the action label from a transition.

Intuitively, a CTS evolves as follows: In the beginning, a version of the system $\varphi \in \Phi$ is chosen and the CTS is instantiated to the version $\varphi$ as the traditional labelled transition system that has a transition $x \xrightarrow{a} x'$ if and only if the CTS has a transition $x \xrightarrow{a,\varphi} x'$. At any point of the execution of this labelled transition system, an upgrade may be performed, i.e., a new version $\varphi'$ with $\varphi' \leq_\Phi \varphi$ of the system may be chosen. The system remains in the state reached up to that point and additional transitions get activated, since now all transitions $x \xrightarrow{a,\varphi'} x'$ give rise to a transition $x \xrightarrow{a} x'$. Note that due to the monotonicity of the transition function $f$ in a CTS, an upgrade will always retain all previous transitions, but may add additional transitions. Symbolically, if $x \xrightarrow{a,\varphi} x'$ and $\varphi' \leq_\Phi \varphi$ then $x \xrightarrow{a,\varphi'} x'$.

The notion of behavioural equivalence we are interested in is conditional bisimulation:

**Definition 3.2.** Let $(X, A, \Phi, f)$ be a CTS. Let $f_\varphi(x, a) = f(x, a)(\varphi)$ (for every $\varphi \in \Phi$) denote the labelled transition system induced upon choosing the condition $\varphi$. A *conditional bisimulation* on the given CTS $(X, A, \Phi, f)$ is a family of relations $(R_\varphi)_{\varphi\in\Phi}$ satisfying the following conditions:

- Each $R_\varphi$ is a traditional bisimulation relation on the LTS $f_\varphi\colon X \times A \to \mathcal{P}X$.
- For every $\varphi, \varphi' \in \Phi$ we have $\varphi' \leq_\Phi \varphi \implies R_\varphi \subseteq R_{\varphi'}$.

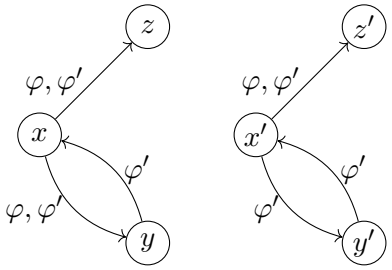
For $x, y \in X$ we say that $x \sim_\varphi y$ if there exists a conditional bisimulation such that $x\, R_\varphi\, y$.

Originally, CTSs were introduced without a notion of upgrades, these systems can be reobtained by setting the order $\leq_\Phi$ on the conditions to be the trivial order.

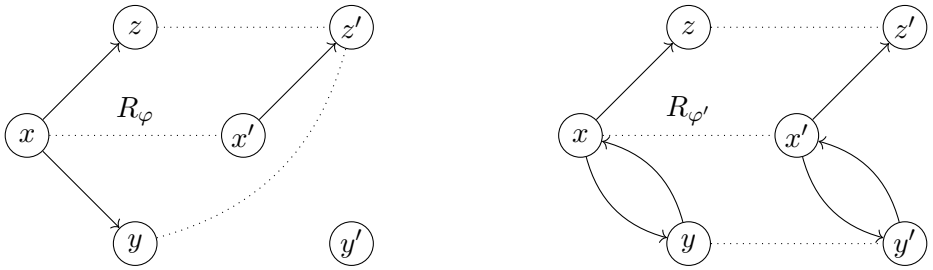
There is a game characterising conditional bisimulation [BKKS17], in which the upgrades are chosen by the attacker, whose aim it is to show that two states are not bisimilar. This also explains Definition 3.2, where we require that $R_\varphi \subseteq R_{\varphi'}$ whenever $\varphi' \leq_\Phi \varphi$. This means that the defender still has a winning strategy after the attacker chooses to make an upgrade.

To get a better feeling of CTSs, consider the following example:

**Example 3.3.** Consider a CTS $(X, \{a\}, \Phi, f)$ as depicted below, where $X = \{x, y, z, x', y', z'\}$ and $\Phi = \{\varphi', \varphi\}$ with $\varphi' \leq_\Phi \varphi$. Since the set of actions is singleton, we leave out the action labels in the visual representation.




We will now detail how the above behavioural description can be represented by a transition function. For instance, the equation $f(x,a)(\varphi) = \{y, z\}$ specifies that the system under the condition $\varphi$ may move nondeterministically from the state $x$ to $y$ or $z$, additionally, it can also upgrade to the condition $\varphi'$.




Consider the labelled transition systems $f_\varphi$ and $f_{\varphi'}$ as depicted above in the left and right, respectively. Notice that the states $x$ and $x'$ are bisimilar in both the instantiations with the relations $R_\varphi$ and $R_{\varphi'}$ depicted as dotted lines. However, we find that $x$ and $x'$ are not conditionally bisimilar, because $y \, R_\varphi \, z'$, but $(y, z') \notin R_{\varphi'}$ and there is no other conditional bisimulation relating $x, x'$. Moreover, the states $y$ and $z'$ in the instantiation $\varphi'$ can never be related by any bisimulation.

The corresponding strategy for the attacker is as follows: start with condition $\varphi$ and make a move from $x$ to $y$. The defender is then forced to take the transition from $x'$ to $z'$. Then the attacker can upgrade to $\varphi'$ and make a move, starting from $y$, which the defender can not mimic in $z'$.

Next, we recall an equivalent, but more compact representation of a CTS which we call lattice transition system (LaTS). In [BKKS17] we showed that behavioural equivalence checks can be performed more efficiently in the lattice setting, by encoding lattice elements into binary decision diagrams.

**Definition 3.4** (Complete Lattice, Frame)**.** A poset $(\mathbb{L}, \leq_\mathbb{L})$ is a *join-complete lattice* if for any subset $L \subseteq \mathbb{L}$ the supremum $\bigsqcup L$ and for any finite subset $L \subseteq \mathbb{L}$, the infimum $\bigsqcap L$ exist.

A *frame* (see e.g. [MLM92]) is a join-complete lattice satisfying the *join-infinite distributive* law:

$$\ell \sqcap \bigsqcup L = \bigsqcup \{\ell \sqcap \ell' \mid \ell' \in L\}, \quad \text{(for any } L \subseteq \mathbb{L}\text{)}. \tag{JID}$$

**Definition 3.5.** A *lattice transition system* (LaTS) over a finite frame $\mathbb{L}$ is a tuple $(X, A, \mathbb{L}, f)$ consisting of a set of states $X$, a set of actions $A$, and a transition function $f\colon X \times A \times X \to \mathbb{L}$.

Even though the frame of a LaTS is required to be finite, and thus is nothing but a finite lattice, the results in the following Section 4 hold for arbitrary frames.

**Remark 3.6.** LaTS can also serve as an explanation why in a CTS, upgrading means going downwards in the partial order. One special case of LaTS arises when choosing $\mathbb{L}$ as the binary Boolean algebra, yielding standard LTS. Using the order and Birkhoff duality as we have done here, the matrix representation of a LaTS over $\{0,1\}$ has the same interpretation as the standard way of writing LTS, i.e., a 1 indicates that a transition is possible, whereas a 0 indicates that no transition is possible. If one were to turn the order around, such that an upgrade means going up in the order, this correspondence gets turned around as well. So in this sense, when LaTS are considered as generalisations of LTS, it is more natural to go down in the order to upgrade, rather than to go up.

**Definition 3.7.** Given a poset $\Phi$, then a subset $\Phi' \subseteq \Phi$ is *downward closed* if

$$\text{for all } \varphi \in \Phi' \text{ and } \psi \leq \varphi, \text{ we have } \psi \in \Phi'.$$

Given a lattice $\mathbb{L}$ with arbitrary joins, $\ell \in \mathbb{L}$ is called *(complete) join irreducible* if $\ell = \bigsqcup L$ for $L \subseteq \mathbb{L}$ implies $\ell \in L$.

**Notation 3.8.** We write $\mathcal{O}(\Phi)$ and $\mathcal{J}(\mathbb{L})$ to denote the set of downward closed subsets of $\Phi$ and the set of join irreducible elements of $\mathbb{L}$, respectively.

As worked out in [BKKS17], a CTS $(X, A, \Phi, f)$ corresponds to a LaTS $(X, A, \mathbb{L}, g)$ where $\mathbb{L} = \mathcal{O}(\Phi)$ and $g\colon X \times A \times X \to \mathbb{L}$ with $g(x,a,x') = \{\varphi \in \Phi \mid x' \in f(x,a)(\varphi)\}$ for $x, x' \in X$, $a \in A$. Similarly, a LaTS can be converted into a CTS by using the Birkhoff duality and by taking the join irreducibles as conditions.

**Remark 3.9.** $\mathcal{O}$ can be defined equivalently as the contravariant hom functor $\mathcal{O} := \mathsf{Poset}(\_, \mathbb{2})\colon \mathsf{Poset}^{\mathsf{op}} \to \mathsf{Frames}$, where $\mathbb{2}$ is the poset/lattice on $\{0,1\}$ with $0 \leq 1$. Similarly, $\mathcal{J}$ is the contravariant hom functor $\mathsf{Frames}(\_, \mathbb{2})\colon \mathsf{Frames}^{\mathsf{op}} \to \mathsf{Poset}$. Taking the respective subcategories of finite posets, resp. frames, the functors $\mathcal{O}$ and $\mathcal{J}$ form an equivalence of categories, known as Birkhoff's theorem:

**Theorem 3.10** (Birkhoff's representation theorem, [DP02, 5.12],[Bir37])**.** *Let $\mathbb{L}$ be a finite frame, then $(\mathbb{L}, \sqcup, \sqcap) \cong (\mathcal{O}(\mathcal{J}(\mathbb{L})), \cup, \cap)$ via the isomorphism $\eta\colon \mathbb{L} \to \mathcal{O}(\mathcal{J}(\mathbb{L}))$, defined as $\eta(\ell) = \{\ell' \in \mathcal{J}(\mathbb{L}) \mid \ell' \sqsubseteq \ell\}$. Furthermore, given a finite poset $(\Phi, \leq_\Phi)$, the downward-closed subsets of $\Phi$, $(\mathcal{O}(\Phi), \cup, \cap)$ form a frame, with inclusion $(\subseteq)$ as the partial order. The irreducibles of this frame are all sets of the form $\downarrow\varphi = \{\varphi' \mid \varphi' \leq_\Phi \varphi\}$ for $\varphi \in \Phi$.*

*Going from $\mathbb{L}$ to the isomorphic $\mathcal{O}(\mathcal{J}(\mathbb{L}))$, each frame element $\ell \in \mathbb{L}$ is mapped to the set of all irreducible elements that are smaller than $\ell$, i.e. $\{\ell' \in \mathcal{J}(\mathbb{L}) \mid \ell' \sqsubseteq \ell\}$.*

Consequently, a LaTS evolves just like a CTS for $\Phi := \mathcal{J}(\mathbb{L})$. At a state and in a version $\ell \in \mathcal{J}(\mathbb{L})$, all the transitions that carry a label of at least $\ell$ remain active, whereas all other transitions are deactivated. At any point of the execution, an upgrade to a smaller join-irreducible element $\ell'$ may be performed, activating additional transitions accordingly. A CTS and a LaTS can be transformed into one another by going from the lattice to its dual partial order and vice-versa (see Section 4). More instructively, the CTS defined in Example 3.3 can be turned into a LaTS by simply writing the conditions inside curly braces and considering those as elements of $\mathcal{O}(\Phi)$.

A benefit of LaTS over CTS is that now bisimulation can be stated in more traditional terms. In addition, this view is also helpful in computing the largest conditional bisimilarity via matrix multiplication (see [BKKS17] for more details).

**Definition 3.11.** Let $(X, A, \mathbb{L}, f)$ be a LaTS and let $\mathcal{J}(\mathbb{L})$ denote the set of all join-irreducible elements of $\mathbb{L}$. A function $R\colon X \times X \to \mathbb{L}$ is a *lattice bisimulation* if and only if the following transfer properties are satisfied.

(1) For all $x, x', y \in X$ $a \in A$, $\ell \in \mathcal{J}(\mathbb{L})$ whenever $x \xrightarrow{a,\ell} x'$ and $\ell \leq_{\mathbb{L}} R(x, y)$, there exists $y' \in X$ such that $y \xrightarrow{a,\ell} y'$ and $\ell \leq_{\mathbb{L}} R(x', y')$.
(2) Symmetric to (1) with the roles of $x$ and $y$ interchanged.

Here, we write $x \xrightarrow{a,\ell} x'$, whenever $\ell \leq_{\mathbb{L}} f(x, a, x')$.

**Theorem 3.12** [BKKS17]**.** *Two states are conditionally bisimilar under condition $\varphi$ if and only if they are related by a lattice bisimulation $R$ with $\varphi \in R(x, y)$.*

## 4.The Lattice Monad

When modelling a LaTS as a coalgebra in the Kleisli category of a monad, the choice of monad is not obvious. One could try to simply use the monad mapping sets to arbitrary lattice-valued functions defined on objects as $TX = \mathbb{L}^X$ and on arrows as $Tf(b)(y) = \bigsqcup_{f(x) \leq_Y y} b(x)$, however, this would not be equivalent to the reader monad. Given a monotone function $f\colon \Phi \to X$, one would like to define a corresponding mapping $\bar{f}\colon X \to \mathbb{L}$ with $\mathbb{L} = \mathcal{O}(\Phi)$ and $\bar{f}(x) = \bigsqcup\{\varphi \in \Phi \mid f(\varphi) \leq x\}$. However, this does not result in a bijection, since some arrows $\bar{f}\colon X \to \mathbb{L}$ do not represent a monotone function $f\colon \Phi \to X$. Hence, we start by imposing restrictions on mappings $\mathbb{L}^X$ and defining a suitable endofunctor in our base category $\mathsf{Poset}$.

Throughout this section, we consider $\mathbb{L}$ to be an arbitrary frame.

**Definition 4.1.** For an ordered set $(X, \leq_X)$, define the poset $TX = (X \to \mathbb{L})^* \subseteq \mathsf{Poset}(X, \mathbb{L})$ as the subset containing all those monotone maps $b\colon X \to \mathbb{L}$ such that for any join-irreducible element $\ell \in \mathcal{J}(\mathbb{L})$, the minimum of $\{x \mid \ell \leq_{\mathbb{L}} b(x)\}$ exists. This means:

$$\exists_{x \in X}\ \ell \leq_{\mathbb{L}} b(x) \wedge \forall_{x' \in X}\ \big(\ell \leq_{\mathbb{L}} b(x') \implies x \leq_X x'\big). \tag{4.1}$$

For functions $b, c \in (X \to \mathbb{L})^*$ we let

$$b \leq_{TX} c \iff \forall_{x \in X}\ b(x) \geq_{\mathbb{L}} c(x).$$

Before stating $T$ as a functor, we canonically relate the function spaces $(X \to \mathbb{L})^*$ and $X^{\mathcal{J}(\mathbb{L})}$.

**Lemma 4.2.** *For each $X$ in $\mathsf{Poset}$, we have a monotone $\tau_X\colon (X \to \mathbb{L})^* \to X^{\mathcal{J}(\mathbb{L})}$ defined by*

$$\tau_X(b)(\ell) = \mathsf{min}\{x \in X \mid \ell \leq_{\mathbb{L}} b(x)\}. \tag{4.2}$$

*Proof.* Given $b \in (X \to \mathbb{L})^*$ and $\ell \in \mathcal{J}(\mathbb{L})$, the minimum $\tau_X(b)(\ell)$ exists.

- Since the minimum is unique if it exists, $\tau_X(b)$ is a map.
- The map $\tau_X(b)\colon \mathcal{J}(\mathbb{L}) \to X$ is monotone, because for $\ell_1 \leq_{\mathbb{L}} \ell_2 \in \mathcal{J}(\mathbb{L})$ with $x_1 := \tau_X(b)(\ell_1)$, and $x_2 := \tau_X(b)(\ell_2)$ we have $\ell_1 \leq_{\mathbb{L}} \ell_2 \leq_{\mathbb{L}} b(x_2)$ and thus $x_1 \leq x_2$ by (4.1) (for $x = x_1$, $x' = x_2$).
- The map $\tau_X$ is monotone in $b \in (X \to \mathbb{L})^*$, because for $b \leq_{TX} c$ and $\ell \in \mathbb{L}$ we have:

$$\begin{array}{rrcl} & \forall_{x \in X}\ b(x) & \geq_{\mathbb{L}} & c(x) \\ \implies & \{x \in X \mid \ell \leq_{\mathbb{L}} b(x)\} & \supseteq & \{x \in X \mid \ell \leq_{\mathbb{L}} c(x)\} \\ \implies & \mathsf{min}\{x \in X \mid \ell \leq_{\mathbb{L}} c(x)\} & \leq_X & \mathsf{min}\{x \in X \mid \ell \leq_{\mathbb{L}} b(x)\} \\ \implies & \tau_X(c)(\ell) & \leq_X & \tau_X(b)(\ell) \end{array}$$

□

**Lemma 4.3.** *We have an adjunction-style situation with $b$ and $\tau_X(b)$, namely*

$$\tau_X(b)(\ell) \leq_X x \quad \textit{iff} \quad \ell \leq_{\mathbb{L}} b(x) \quad \textit{for all } x \in X, \ell \in \mathcal{J}(\mathbb{L}) \tag{4.3}$$

*Proof.* The direction $\Rightarrow$ holds because by definition of $\tau$, $\ell \leq_{\mathbb{L}} b(\tau_X(b)(\ell))$ and so $\ell \leq b(x)$ by monotonicity of $b$. For $\Leftarrow$, recall that $\tau_X(b)(\ell)$ is the least element in $X$ with $\ell \leq_{\mathbb{L}} b(x)$. □

This correspondence is not a proper adjunction (or in $\mathsf{Poset}$ equivalently a Galois connection), because $\tau_X(b)$ is only defined for $\ell \in \mathcal{J}(\mathbb{L})$ and not for all elements of $\mathbb{L}$.

**Lemma 4.4.** *$\tau_X$ is an isomorphism; its inverse $\tau_X^{-1} \colon X^{\mathcal{J}(\mathbb{L})} \to (X \to \mathbb{L})^*$ is given by*

$$\tau_X^{-1}(B)(x) = \bigsqcup \{\ell \in \mathcal{J}(\mathbb{L}) \mid B(\ell) \leq_X x\}$$

*and for $B \colon \mathcal{J}(\mathbb{L}) \to X$,*

$$\ell \leq_{\mathbb{L}} \tau_X^{-1}(B)(x) \quad \textit{iff} \quad B(\ell) \leq_X x \quad \textit{for all } x \in X, \ell \in \mathcal{J}(\mathbb{L}). \tag{4.4}$$

*Proof.*

- First of all $\tau_X^{-1}(B) \colon X \to \mathbb{L}$ is a monotone map, because if $x \leq_X x'$, then
$$\tau_X^{-1}(B)(x) = \bigsqcup \{\ell \in \mathcal{J}(\mathbb{L}) \mid B(\ell) \leq x\} \leq_{\mathbb{L}} \bigsqcup \{\ell \in \mathcal{J}(\mathbb{L}) \mid B(\ell) \leq x'\} = \tau_X^{-1}(B)(x').$$
- For $B \colon \mathcal{J}(\mathbb{L}) \to X$, we have (4.4) for all $\ell \in \mathcal{J}(\mathbb{L})$ and $x \in X$, because:
$$\begin{aligned}
\ell \leq_{\mathbb{L}} \tau_X^{-1}(B)(x) &\iff \ell \leq_{\mathbb{L}} \bigsqcup \{\ell' \in \mathcal{J}(\mathbb{L}) \mid B(\ell') \leq_X x\} \\
&\iff \ell = \ell \sqcap \bigsqcup \{\ell' \in \mathcal{J}(\mathbb{L}) \mid B(\ell') \leq_X x\} \\
&\overset{\mathsf{JID}}{\iff} \ell = \bigsqcup \{\ell \sqcap \ell' \mid \ell' \in \mathcal{J}(\mathbb{L}),\ B(\ell') \leq_X x\} \\
&\overset{\ell \in \mathcal{J}(\mathbb{L})}{\iff} \ell \in \{\ell \sqcap \ell' \mid \ell' \in \mathcal{J}(\mathbb{L}),\ B(\ell') \leq_X x\} \\
&\iff \exists_{\ell' \in \mathcal{J}(\mathbb{L})}\ \ell = \ell \sqcap \ell' \text{ and } B(\ell') \leq_X x \\
&\iff \exists_{\ell' \in \mathcal{J}(\mathbb{L})}\ \ell \leq_{\mathbb{L}} \ell' \text{ and } B(\ell') \leq_X x \\
&\overset{B \text{ monotone}}{\iff} B(\ell) \leq x
\end{aligned}$$
- $\tau^{-1}$ is monotone in $B$, because for any $B \leq C$ and $x \in X$
$$\begin{array}{rrcl}
 & \forall_{\ell \in \mathbb{L}} \quad B(\ell) & \leq_X & C(\ell) \\
\implies & \{\ell \in \mathcal{J}(\mathbb{L}) \mid B(\ell) \leq_X x\} & \supseteq & \{\ell \in \mathcal{J}(\mathbb{L}) \mid C(\ell) \leq_X x\} \\
\implies & \bigsqcup \{\ell \in \mathcal{J}(\mathbb{L}) \mid B(\ell) \leq_X x\} & \geq_{\mathbb{L}} & \bigsqcup \{\ell \in \mathcal{J}(\mathbb{L}) \mid C(\ell) \leq_X x\} \\
\implies & \tau_X^{-1}(B)(x) & \geq_{\mathbb{L}} & \tau_X^{-1}(C)(x)
\end{array}$$
and so $\tau_X^{-1}(B) \leq_{TX} \tau_X^{-1}(C)$.
- For $B \colon \mathcal{J}(\mathbb{L}) \to X$, $b := \tau_X^{-1}(B)$, and $\ell \in \mathcal{J}(\mathbb{L})$ the witness for (4.1) is $B(\ell) \in X$:
$$\mathsf{min}\{x \in X \mid \ell \leq_{\mathbb{L}} b(x)\} \overset{(4.4)}{=} \mathsf{min}\{x \in X \mid B(\ell) \leq_X x\} = B(\ell) \tag{4.5}$$
So $B(\ell)$ is the desired witness for (4.1).
- We have $\tau_X(\tau_X^{-1}(B)) = B$ by (4.5).
- For the converse, if $b \in (X \to \mathbb{L})^*$ then we have for all $x \in X$:
$$\begin{aligned}
\tau_X^{-1}(\tau_X(b))(x) &= \bigsqcup \{\ell \in \mathcal{J}(\mathbb{L}) \mid \tau_X(b)(\ell) \leq_X x\} \\
&\overset{(4.3)}{=} \bigsqcup \{\ell \in \mathcal{J}(\mathbb{L}) \mid \ell \leq_{\mathbb{L}} b(x)\} \ = \ b(x)
\end{aligned}$$
□

So we now have an object mapping $T\colon \mathbf{obj}\,\mathsf{Poset} \to \mathbf{obj}\,\mathsf{Poset}$ and a family of isomorphisms $\tau_X\colon TX \cong X^{\mathcal{J}(\mathbb{L})}$. Since $\_^{\mathcal{J}(\mathbb{L})}$ is already a functor, this induces a mapping on monotone maps for $T$:

**Definition 4.5.** Define $T$ on monotone maps $f\colon X \to Y$ by

$$Tf(b) := \tau_Y^{-1} \cdot f^{\mathcal{J}(\mathbb{L})} \cdot \tau_X(b) \quad \text{for } b \in (X \to \mathbb{L})^*$$

making $T$ a functor.

$$\begin{array}{ccc} (X \to \mathbb{L})^* & \xrightarrow{\tau_X} & X^{\mathcal{J}\mathbb{L}} \\ \downarrow Tf := & & \downarrow f^{\mathcal{J}\mathbb{L}} \\ (Y \to \mathbb{L})^* & \underset{\tau_Y}{\overset{\tau_Y^{-1}}{\leftrightarrows}} & Y^{\mathcal{J}\mathbb{L}} \end{array}$$

**Remark 4.6.** Using that $\_^{\mathcal{J}(\mathbb{L})}$ is a functor, $T$ automatically preserves identities and composition. So by definition, $T\colon \mathsf{Poset} \to \mathsf{Poset}$ is a functor and $\tau\colon T \to \_^{\mathcal{J}(\mathbb{L})}$ is a natural isomorphism.

**Proposition 4.7.** *For $f\colon X \to Y$, $b \in (X \to \mathbb{L})^*$, $y \in Y$ we have*

$$Tf(b)(y) = \bigsqcup_{f(x) \leq_Y y} b(x)$$

*Proof.*

$$\begin{aligned} Tf(b)(y) = \big((\tau_Y^{-1} \cdot f^{\mathcal{J}(\mathbb{L})} \cdot \tau_X)(b)\big)(y) &= \bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid \overbrace{f(\tau_X(b)(\ell))}^{((f^{\mathcal{J}(\mathbb{L})} \cdot \tau_X)(b))(\ell)} \leq y\} \\ \overset{f \text{ monotone}}{=} &\bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid x \in X, \tau_X(b)(\ell) \leq x, f(x) \leq y\} \\ \overset{(4.3)}{=} &\bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid x \in X, \ell \leq b(x), f(x) \leq y\} \\ \overset{\sqcup \text{ associative}}{=} &\bigsqcup\big\{\bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid \ell \leq b(x)\} \mid x \in X, f(x) \leq y\big\} \\ = &\bigsqcup\big\{b(x) \mid x \in X, f(x) \leq y\big\} \end{aligned}$$

□

Using the same pattern as in Definition 4.5, $T$ carries a canonical monad structure:

**Definition 4.8.** Define the monad structure $\eta\colon \mathrm{Id} \to T$, $\mu\colon TT \to T$ on $T\colon \mathsf{Poset} \to \mathsf{Poset}$ by

$$\begin{array}{ccc} X & & \\ & \overset{\nu_X}{\searrow} & \\ \eta_X \downarrow := & & \\ TX & \underset{\tau_X}{\overset{\tau_X^{-1}}{\leftrightarrows}} & X^{\mathcal{J}(\mathbb{L})} \end{array} \qquad \begin{array}{ccc} TTX & \xrightarrow{(\tau * \tau)_X} & (X^{\mathcal{J}(\mathbb{L})})^{\mathcal{J}(\mathbb{L})} \\ \mu_X \downarrow := & & \downarrow \zeta_X \\ TX & \underset{\tau_X}{\overset{\tau_X^{-1}}{\leftrightarrows}} & X^{\mathcal{J}(\mathbb{L})} \end{array}$$

Here $\tau * \tau\colon TT \to (\_^{\mathcal{J}(\mathbb{L})})^{\mathcal{J}(\mathbb{L})}$ is the Godement product (or star product, or horizontal composition), defined by $(\tau * \tau)_X := (\tau_X)^{\mathcal{J}(\mathbb{L})} \cdot \tau_{TX} = \tau_{X^{\mathcal{J}(\mathbb{L})}} \cdot T\tau_X$ (naturally equivalent).

Again trivially, $\eta$ and $\mu$ are natural transformations because $\tau$, $\nu$, and $\zeta$ are, and furthermore fulfill the monad laws, because $\nu$ and $\zeta$ do. By definition, $\tau$ is a monad isomorphism.

**Proposition 4.9.** *Explicitly speaking, the monad structure on $T$ is defined as follows:*

$$\eta_X(x)(x') = \begin{cases} \top & \text{if } x \leq x' \\ \bot & \text{otherwise} \end{cases}$$

$$\mu_X(h)(x) = \bigsqcup_{b \in (X \to \mathbb{L})^*} (h(b) \sqcap b(x)), \qquad \text{where } h \in ((X \to \mathbb{L})^* \to \mathbb{L})^*.$$

*Proof.* For the unit $\eta_X \colon X \to (X \to \mathbb{L})^*$ and $x, x' \in X$ we have directly:

$$\eta_X(x)(x') = (\tau_X^{-1} \cdot \nu_X(x))(x') = \bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid \nu_X(x)(\ell) \leq x'\} = \bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid x \leq x'\}$$
$$= \begin{cases} \bigsqcup_{\ell \in \mathcal{J}(\mathbb{L})} \ell & \text{if } x \leq x' \\ \bigsqcup \emptyset & \text{otherwise} \end{cases} = \begin{cases} \top & \text{if } x \leq x' \\ \bot & \text{otherwise} \end{cases}$$

Before characterising $\mu$, we first prove that for all $h \in ((X \to \mathbb{L})^* \to \mathbb{L})^*$ and $x \in X$,

$$\ell \leq_{\mathbb{L}} \bigsqcup_{b \in (X \to \mathbb{L})^*} h(b) \sqcap b(x) \quad \Longleftrightarrow \quad \ell \leq_{\mathbb{L}} \tau_{TX}(h)(\ell)(x) \qquad \text{for all } \ell \in \mathcal{J}(\mathbb{L}). \tag{4.6}$$

($\Rightarrow$) Note that for any $L \subseteq \mathbb{L}$, if $\ell \leq_{\mathbb{L}} \bigsqcup L$, then $\ell = \ell \sqcap \bigsqcup L = \bigsqcup_{\ell' \in L} \ell \sqcap \ell'$ (using $\mathsf{JID}$), and since $\ell$ is join-irreducible, there is some $\ell' \in L$ with $\ell \leq_{\mathbb{L}} \ell'$. Hence for the current assumption, there is some $b \in (X \to \mathbb{L})^*$ with $\ell \leq_{\mathbb{L}} h(b) \sqcap b(x)$. Since in particular $\ell \leq_{\mathbb{L}} h(b)$, we have $\tau_{TX}(h)(\ell) \leq_{TX} b$ by (4.3) and finally $\ell \leq_{\mathbb{L}} b(x) \leq_{\mathbb{L}} \tau_{TX}(h)(\ell)(x)$ by the definition of $\leq_{TX}$.

($\Leftarrow$) For $b := \tau_{TX}(h)(\ell) = \mathsf{min}\{c \in TX \mid \ell \leq_{\mathbb{L}} h(c)\}$ we have $\ell \leq_{\mathbb{L}} h(b)$ by the definition of $\tau$ and $\ell \leq_{\mathbb{L}} b(x)$ by the current assumption; hence $\ell \leq_{\mathbb{L}} h(b) \sqcap b(x)$.

Now for $h \in TTX, x \in X$, $\mu_X \colon ((X \to \mathbb{L})^* \to \mathbb{L})^* \to (X \to \mathbb{L})^*$ is characterised as desired:

$$\begin{aligned}
(\mu_X(h))(x) &= \big((\tau_X^{-1} \cdot \zeta_X \cdot (\tau * \tau)_X)(h)\big)(x) \\
&= \big((\tau_X^{-1} \cdot \zeta_X \cdot \tau_X^{\mathcal{J}(\mathbb{L})} \cdot \tau_{TX})(h)\big)(x) \\
&\overset{\text{Def}}{=} \bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid \big((\zeta_X \cdot \tau_X^{\mathcal{J}(\mathbb{L})} \cdot \tau_{TX})(h)\big)(\ell) \leq_X x\} \\
&= \bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid \tau_X(\underbrace{\tau_{TX}(h)(\ell)}_{\in TX})(\ell) \leq_X x\} \\
&\overset{(4.3)}{=} \bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid \ell \leq_{\mathbb{L}} \tau_{TX}(h)(\ell)(x)\} \\
&\overset{(4.6)}{=} \bigsqcup\{\ell \in \mathcal{J}(\mathbb{L}) \mid \ell \leq_{\mathbb{L}} \bigsqcup_{b \in TX} (h(b) \sqcap b(x))\} = \bigsqcup_{b \in (X \to \mathbb{L})^*} (h(b) \sqcap b(x)).
\end{aligned}$$

□

It is a standard exercise to see that there is a one-to-one correspondence between monad morphisms and functors between their Kleisli categories [Mog89, Prop. 4.0.10]. So $\tau$ induces an isomorphism between categories $\mathsf{Kl}(T) \xrightarrow{\cong} \mathsf{Kl}(\_^{\mathcal{J}(\mathbb{L})})$, defined as

$$(f \colon X \to TY) \mapsto (\tau_X \cdot f \colon X \to Y^\Phi).$$

Now when fixing a *finite* partially ordered set $\Phi$ and putting $\mathbb{L} := \mathcal{O}(\Phi)$, Birkhoff's theorem (cf. Theorem 3.10) provides $\Phi \cong \mathcal{J}(\mathbb{L})$ and so $T \cong \_^{\mathcal{J}(\mathbb{L})} \cong \_^\Phi$.

## 5. Modelling Conditional Transition Systems as Coalgebras

Recall that once a condition is fixed by a CTS then it behaves like a traditional transition system (until another upgrade). Thus, it is natural to consider the powerset functor to model the set of successor states when the upgrade order is discrete. This way of modelling CTS adapts the approach in [ABH+12], where the set of actions $A$ was fixed to be singleton.

**Definition 5.1.** The powerset functor $\mathcal{P}$ on $\mathsf{Poset}$ maps posets $(X, \leq)$ to $(\mathcal{P}X, \subseteq)$, the subsets of $X$ ordered by inclusion. For $f \colon X \to Y$, $\mathcal{P}f(S) = f[S]$ is the forward image.

**Remark 5.2.** In other words, $\mathcal{P}$ on $\mathsf{Poset}$ is the composition of the forgetful functor $\mathsf{Poset} \to \mathsf{Set}$ with the ordinary powerset $\mathsf{Set} \to \mathsf{Poset}$. Sometimes, the dual functor $\mathcal{D}\colon \mathsf{Poset} \xrightarrow{\cong} \mathsf{Poset}$ is required which sends each poset $(X, \leq_X)$ to its dual $(X, \geq_X)$. Then, the composition $\mathcal{DP}(X, \leq_X)$ contains the subsets of $X$ ordered by inverse inclusion.

Next, we define two functors $H\colon \mathsf{Poset} \to \mathsf{Poset}$ – based on $\mathcal{P}$ – for modelling CTS as coalgebras for the extension of $H$ to the Kleisli category $\mathsf{Kl}(\_^{\Phi})$. The first one closely follows the concrete Definition 3.1 with the reversed inclusion order. However, it turns out that this functor can not be extended to the Kleisli category for non-discrete $\Phi$ (cf. Example 5.8). Hence, we also consider a second functor, which not only records all the successors for a given condition $\varphi$, but also all possible successors for conditions $\varphi' \leq \varphi$ via pairs of the form $(x, \varphi)$. In order to faithfully model CTSs, we here need to consider the usual inclusion order (if a condition is larger we have more potential upgrades).

**Remark 5.3.** A CTS $(X, A, \Phi, f)$ defines the following Kleisli morphisms:

(1) Considering the sets $X$ and $A$ as discrete posets $(X, =)$ and $(A, =)$, the map $f\colon X \times A \to \mathsf{Poset}(\Phi, \mathcal{DP}X)$ is a morphism

$$f\colon X \times A \longrightarrow (\mathcal{DP}X)^{\Phi} \qquad \text{in } \mathsf{Poset}.$$

Up to exponential laws, this corresponds to

$$\alpha\colon X \longrightarrow ((\mathcal{DP}X)^{A})^{\Phi} \qquad \text{in } \mathsf{Poset},$$

in other words a Kleisli morphism

$$\alpha\colon X \mathrel{-\!\!\circ\!\!\to} (\mathcal{DP}X)^{A} \qquad \text{in } \mathsf{Kl}(\_^{\Phi}).$$

However, this is not necessarily a coalgebra, since we do not have an endofunctor on $\mathsf{Kl}(\_^{\Phi})$ yet. In the following, an extension of $\mathcal{DP}(\_)^{A}$ is provided for discrete $\Phi$. Furthermore, it is shown that there is no meaningful extension for non-discrete $\Phi$. For discrete $\Phi$, the order does not make a difference, so we can model CTS as $\mathcal{P}$-coalgebras

$$\alpha\colon X \mathrel{-\!\!\circ\!\!\to} (\mathcal{P}X)^{A} \qquad \text{in } \mathsf{Kl}(\_^{\Phi}) \text{ for discrete } \Phi. \tag{5.1}$$

(2) Another way is to encode the possible upgrades explicitly in the morphism. Therefore, define the monotone map $\alpha\colon X \to (\mathcal{P}(X \times \Phi)^{A})^{\Phi}$ directly by

$$\alpha(x)(\varphi)(a) = \{(x', \varphi') \mid x \xrightarrow{a,\varphi'} x' \wedge \varphi' \leq \varphi\}. \tag{5.2}$$

By the discreteness of $X$ and $A$, $\alpha$ is trivially monotone in $x$ and $a$. For $\psi \leq \varphi$,

$$\begin{aligned}(x', \varphi') \in \alpha(x)(\psi)(a) &\Rightarrow x \xrightarrow{a,\varphi'} x' \text{ and } \varphi' \leq \psi \\ &\Rightarrow x \xrightarrow{a,\varphi'} x' \text{ and } \varphi' \leq \varphi \Rightarrow (x', \varphi') \in \alpha(x)(\varphi)(a)\end{aligned}$$

so $\alpha$ is monotone in $\varphi$. As for the previous functor, we can read (5.2) as a Kleisli arrow

$$\alpha\colon X \mathrel{-\!\!\circ\!\!\to} \mathcal{P}(X \times \Phi)^{A}$$

which is a coalgebra as soon as an extension of $\mathcal{P}(\_ \times \Phi)^{A}$ to $\mathsf{Kl}(\_^{\Phi})$ is provided.

5.1.**Functor Extensions.** Independently from $\mathsf{Poset}$, functor extensions to the Kleisli category of the reader monad are of special shape: it is just a tensorial strength fixing of one parameter that fulfills two axioms. Actually, we state our characterisation in an even higher generality by recognising that the monad structure on an endofunctor $T$ is induced by a comonad $L$ when $L \dashv T$ (cf. Proposition 2.5). This characterisation is afterwards used to extend the two functors for CTS to $\mathsf{Kl}(\_^{\Phi})$.

**Lemma 5.4.** *Recall from Proposition 2.5 that a comonad $(L, \vartheta, \delta)$ induces a monad $(T, \eta, \mu)$ when $L \dashv T$ with the unit and the counit of adjunctions as $\rho$ and $\epsilon$, respectively. Then, there is a one-to-one correspondence between distributive laws $\lambda\colon HT \to TH$ and comonad-over-functor distributive laws $\Lambda\colon LH \to HL$.*

*Proof.* The Kleisli category $\mathsf{Kl}(T)$ is isomorphic to the co-Kleisli category $\mathsf{coKl}(L)$ of $L$:

$$\mathsf{Kl}(T)(X,Y) \cong \mathcal{C}(X,TY) \cong \mathcal{C}(LX,Y) \cong \mathsf{coKl}(T)(X,Y)$$

So we have a one to one correspondence between extensions $\hat{H}$ of $H$ to $\mathsf{Kl}(T)$ and extensions $\tilde{H}$ of $H$ to $\mathsf{coKl}(L)$:

$$\begin{array}{ccc} \mathsf{Kl}(T) & \xrightarrow{\hat{H}} & \mathsf{Kl}(T) \\ {\scriptstyle I}\uparrow & & \uparrow{\scriptstyle I} \\ \mathcal{C} & \xrightarrow{H} & \mathcal{C} \end{array} \quad \Longleftrightarrow \quad \begin{array}{ccc} \mathsf{coKl}(L) & \xrightarrow{\tilde{H}} & \mathsf{coKl}(L) \\ {\scriptstyle \tilde{I}}\uparrow & & \uparrow{\scriptstyle \tilde{I}} \\ \mathcal{C} & \xrightarrow{H} & \mathcal{C} \end{array}$$

where $\tilde{I}(f\colon X \to Y) = f \cdot \vartheta\colon LX \to Y$ is just the uncurrying of $I(f)$. Recall from Theorem 2.10 that such extensions $\hat{H}$ are in one-to-one correspondence to distributive laws of $H$ over the monad $T$, and dually are such extensions $\tilde{H}$ in one-to-one correspondence to distributive laws $\Lambda\colon LH \to HL$. □

**Remark 5.5.** Concretely, $\Lambda$ defines a distributive law $\lambda$ by composition

$$\lambda_X :\equiv \big(HTX \xrightarrow{\bar{\Lambda}_{TX}} THLTX \xrightarrow{TH\epsilon_X} THX\big).$$

The functor extension $\hat{H}\colon \mathsf{Kl}(T) \to \mathsf{Kl}(T)$ is then defined as follows, for an $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$ and its corresponding $\check{f}\colon LX \to Y$:

$$\hat{H}(f\colon X \mathrel{-\!\!\circ\!\!\to} Y) :\equiv \big(HX \xrightarrow{\bar{\Lambda}_X} THLX \xrightarrow{TH\check{f}} THY\big) \tag{5.3}$$

We can now apply this to the comonad $\_ \times \Phi$ and the monad $\_^{\Phi}$ on $\mathsf{Poset}$.

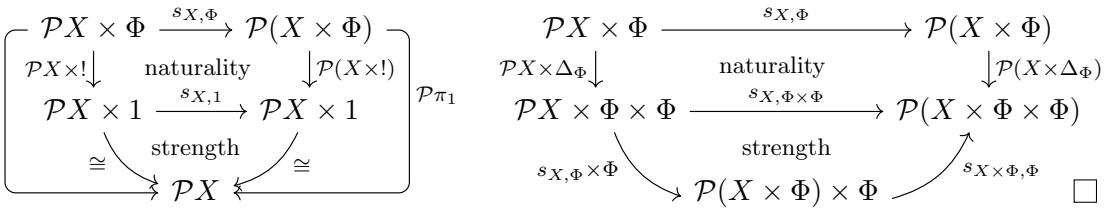
**Definition 5.6.** For a discrete poset $\Phi$, the tensorial strength of $\mathcal{P}$ on $\mathsf{Set}$ defines a family of monotone maps:

$$p_X\colon \mathcal{P}X \times \Phi \to \mathcal{P}(X \times \Phi), \quad p_X(C,\varphi) := \{(x,\varphi) \mid x \in C\}.$$

By the naturality in $\mathsf{Set}$, $p$ is a natural transformation in $\mathsf{Poset}$. And since $\Phi$ is discrete, $p$ is monotone in $\Phi$.

**Lemma 5.7.** *The above $p$ is a distributive law of the comonad $\_ \times \Phi$ over $\mathcal{P}$.*

*Proof.* Using the axioms of the strength $s_{X,Y}$ of $\mathcal{P}\colon \mathsf{Set} \to \mathsf{Set}$, the following diagrams in $\mathsf{Set}$ prove that $p_X = t_{X,\Phi}$ is a distributive law:




Now to model CTS with action labels (the case when $|A| > 1$) we use a distributive law between the functor $\_^A$ and $\_^\Phi$. Recall from [AHS90, Prop 27.8(1)], that any cartesian closed category has the power law $(\_^\Phi)^A \cong (\_^A)^\Phi$. Any natural isomorphism is a distributive law, and so is $\iota\colon (\_^\Phi)^A \to (\_^A)^\Phi$.

We have now completely defined the functor $\hat{\mathcal{P}^A}$ on the Kleisli category. A Kleisli arrow $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$, i.e., $f\colon X \to Y^\Phi$ is mapped to the Kleisli arrow

$$(\mathcal{P}X)^A \xrightarrow{(\mathcal{P}f)^A} \mathcal{P}(Y^\Phi)^\Phi \xrightarrow{\lambda^A_Y} ((\mathcal{P}Y)^\Phi)^A \xrightarrow{\iota_{\mathcal{P}Y}} ((\mathcal{P}Y)^A)^\Phi.$$

As a result, the Kleisli arrow $\alpha\colon X \mathrel{-\!\!\circ\!\!\to} \mathcal{P}(X)^A$ (5.1) induced by a CTS is indeed a coalgebra on $\mathsf{Kl}(\_^\Phi)$ for discrete $\Phi$.

In the case of a non-discrete $\Phi$, the above $\bar{p}$ is not defined since $p_X$ is not necessarily order preserving (even for discrete poset $X$). But, more generally, it is not possible to extend $\mathcal{P}$ to $\mathsf{Kl}(\_^\Phi)$ with the right notion of behavioural equivalence.

**Example 5.8.** Consider the set of conditions $\Phi = \{\varphi, \varphi'\}$ with $\varphi' \leq \varphi$ and a singleton set of actions $A = \{*\}$. Define the CTS $\alpha\colon X \to (\mathcal{D}\mathcal{P}X)^\Phi$ on the discrete $X = \{x_1, x_2\}$

$$\alpha\colon \quad (x_1) \qquad\qquad (x_2) \circlearrowleft \varphi'$$

in equations, $\alpha(x_2)(\varphi') = \{x_2\}$ and $\emptyset$ elsewhere. Then for any extension $\widehat{\mathcal{D}\mathcal{P}}\colon \mathsf{Kl}(\_^\Phi) \to \mathsf{Kl}(\_^\Phi)$ of $\mathcal{D}\mathcal{P}$, $x_1$ and $x_2$ are identified in $\varphi$ by a $\widehat{\mathcal{D}\mathcal{P}}$-coalgebra homomorphism, even though they are not conditionally bisimilar in $\varphi$.

*Proof.* Note that $x_1$ and $x_2$ are not bisimilar under $\varphi'$, because $x_1$ can do a step whereas $x_2$ can not. So there is no bisimulation $R_{\varphi'}$ relating $x_1$ and $x_2$ in $\varphi'$. Consequently, there is no conditional bisimulation with $(x_1, x_2) \in R_\varphi$, since $R_\varphi \subseteq R_{\varphi'}$. However, we can identify $x_1$ and $x_2$ in $\varphi$ by a $\widehat{\mathcal{D}\mathcal{P}}$-coalgebra homomorphism $h\colon (X, \alpha) \mathrel{-\!\!\circ\!\!\to} (Y, \beta)$, where $Y = \{y_1, y_2\}, y_2 \leq y_1$ and where $\beta$ will be defined afterwards:

$$h(x_1)(\varphi') = h(x_1)(\varphi) = h(x_2)(\varphi) = y_1 \qquad\qquad h(x_2)(\varphi') = y_2$$

Since $y_1 \geq y_2$, $h$ is monotone. Having $h$, we can define $\beta$:

$$\beta(y_1)(\varphi') = \beta(y_1)(\varphi) = \beta(y_2)(\varphi) = \emptyset$$
$$\beta(y_2)(\varphi') = (\widehat{\mathcal{D}\mathcal{P}}h \circ \alpha)(x_2)(\varphi') = \widehat{\mathcal{D}\mathcal{P}}h(\alpha(x_2)(\varphi'))(\varphi')$$

Monotonicity of $\beta$ holds in both arguments:

$$\varphi' \leq \varphi \qquad \Longrightarrow \qquad \beta(y_2)(\varphi') \supseteq \beta(y_2)(\varphi) = \emptyset$$
$$y_2 \leq y_1 \qquad \Longrightarrow \qquad \beta(y_2)(\varphi') \supseteq \beta(y_1)(\varphi') = \emptyset$$

It remains to show that $h$ is a coalgebra homomorphism. Recall that in terms of the corresponding distributive law $\lambda$, $\widehat{\mathcal{DP}}h$ is defined as

$$\widehat{\mathcal{DP}}h \equiv \mathcal{DP}X \xrightarrow{\mathcal{DP}h} \mathcal{DP}(Y^\Phi) \xrightarrow{\lambda_Y} \mathcal{DP}Y^\Phi.$$

We know that $\mathcal{DP}h(\emptyset) = \emptyset$, and since $\lambda$ preserves the unit $\nu_Y$,

$$\lambda_Y(\emptyset) = \lambda_Y(\mathcal{DP}\nu_Y(\emptyset)) = \nu_{\mathcal{DP}Y}(\emptyset)$$

and so we have in total that $\widehat{\mathcal{DP}}h(\emptyset) = \nu_{PY}(\emptyset)$. Hence, $h$ is indeed a homomorphism:

$$\begin{array}{ccc} x_1 & \xrightarrow{\alpha_\psi} & \emptyset \\ {\scriptstyle h_\psi}\downarrow & & \downarrow{\scriptstyle (\widehat{\mathcal{DP}}h)_\psi} \\ y_1 & \xrightarrow{\beta_\psi} & \emptyset \end{array} \text{ for all } \psi \in \Phi \qquad \begin{array}{ccc} x_2 & \xrightarrow{\alpha_\varphi} & \emptyset \\ {\scriptstyle h_\varphi}\downarrow & & \downarrow{\scriptstyle (\widehat{\mathcal{DP}}h)_\varphi} \\ y_1 & \xrightarrow{\beta_\varphi} & \emptyset \end{array} \qquad \begin{array}{ccc} x_2 & \xrightarrow{\alpha_{\varphi'}} & \alpha(x_2)(\varphi') \\ {\scriptstyle h_{\varphi'}}\downarrow & & \downarrow{\scriptstyle (\widehat{\mathcal{DP}}h)_{\varphi'}} \\ y_2 & \xrightarrow{\beta_{\varphi'}} & (\hat{\mathcal{P}}h \circ \alpha)(x_2)(\varphi') \end{array} \qquad \square$$

Hence another functor, namely $\mathcal{V} = \mathcal{P}(\_ \times \Phi)$, is necessary to model upgrades:

**Proposition 5.9.** *The* $\mathsf{Poset}$*-functor* $\mathcal{P}(\_ \times \Phi)$ *extends to* $\mathsf{Kl}(\_^\Phi)$ *using the comonad distributive law:*

$$\Lambda_X \colon \mathcal{P}(X \times \Phi) \times \Phi \xrightarrow{\pi_1} \mathcal{P}(X \times \Phi) \xrightarrow{\mathcal{P}(\mathrm{id}_X \times \Delta_\Phi)} \mathcal{P}(X \times \Phi \times \Phi) \tag{5.4}$$

The corresponding distributive law by Lemma 5.4 is

$$\lambda_X \colon \mathcal{P}(X^\Phi \times \Phi) \xrightarrow{\mathcal{P}\langle \mathrm{ev}_X, \pi_2 \rangle} \mathcal{P}(X \times \Phi) \xrightarrow{\nu_{\mathcal{P}(\ldots)}} \mathcal{P}(X \times \Phi)^\Phi$$

*Proof.* The counit $\pi_1$ of $\_ \times \Phi$ is preserved by $\Lambda$, i.e. $\mathcal{P}(\pi_1 \times \Phi) \cdot \Lambda = \pi_1$, because

$$\mathcal{P}(\pi_1 \times \Phi) \cdot \mathcal{P}(\mathrm{id}_X \times \Delta_\Phi) = \mathcal{P}(\mathrm{id}_X \times \mathrm{id}_\Phi)$$

The comultiplication $\delta_X = X \times \Delta_\Phi$ of $\_ \times \Phi$ is preserved because $\Lambda_X = \mathcal{P}(\delta_X) \cdot \pi_1$: of course $\mathcal{P}(\delta_{X \times \Phi}) \cdot \mathcal{P}(\delta_X) = \mathcal{P}(\delta_X \times \Phi) \cdot \mathcal{P}(\delta_X)$, and precomposing this with $\pi_1$ and using naturality we have that $\Lambda$ preserves $\delta$. $\square$

Moreover, this extension can be composed with $(\_^\Phi)^A \cong (\_^A)^\Phi$ to obtain an extension of $\mathcal{P}(\_ \times \Phi)^A$. So $\alpha \colon X \nrightarrow \mathcal{P}(X \times \Phi)^A$ from (5.2) indeed defines a coalgebra on $\mathsf{Kl}(\_^\Phi)$.

**Remark 5.10.** The $\mathsf{Kl}(\_^\Phi)$-extension $\widehat{\mathcal{V}}$ of the $\mathsf{Poset}$-functor $\mathcal{P}(\_ \times \Phi)^A$ has the explicit form: for an arrow $f \colon X \to Y$ in $\mathsf{Kl}(\_^\Phi)$, $\widehat{\mathcal{V}}f \colon (\mathcal{P}(X \times \Phi))^A \to (\mathcal{P}(Y \times \Phi))^A$ is a function $\widehat{\mathcal{V}}f \colon (\mathcal{P}(X \times \Phi))^A \to ((\mathcal{P}(Y \times \Phi))^A)^\Phi$, where

$$\widehat{\mathcal{V}}f(p)(\varphi)(a) = \{(f(x)(\varphi'), \varphi') \mid (x, \varphi') \in p(a)\},$$

for all $p \in (\mathcal{P}(X \times \Phi))^A$, $\varphi \in \Phi$ and $a \in A$.

5.2.**Coalgebraic Behavioural Equivalence.** Having defined coalgebras of interests on the Kleisli category $\mathsf{Kl}(\_^\Phi)$, our next motive is to characterise conditional bisimilarity using the notion of behavioural equivalence in $\mathsf{Kl}(\_^\Phi)$. To this end, we first define a coalgebraic generalisation of conditional bismilarity (Definition 3.2), requiring additional structure on a general functor $H\colon \mathsf{Poset} \to \mathsf{Poset}$, and then prove that this notion coincides with the coalgebraic behavioural equivalence of coalgebras on $\mathsf{Kl}(\_^\Phi)$.

**Definition 5.11** (constant map)**.** For a poset $X$, there is a unique monotone map $!\colon X \to 1$, the *final morphism*. Also note that for any element $x \in X$, one has a monotone map $x\colon 1 \to X$, mapping the only element in 1 to $x \in X$. The composition of the above two maps $x!\colon X \to X$ is the constant map sending any element of $X$ to $x$.

**Remark 5.12.** Recall that a family of morphisms $(f_i\colon Y \to Z_i)_{i\in I}$ is *jointly monic*, if for morphisms $g, h\colon X \to Y$ with $f_i \cdot g = f_i \cdot h$ for all $i \in I$ we have $g = h$. If the category has products, such a family is jointly monic iff $\langle f_i\rangle_{i\in I}\colon Y \to \prod_{i\in I} Z_i$ is monic. For instance, the limit projections form a jointly-monic family.

For a Kleisli arrow $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$ and a condition $\varphi \in \Phi$, recall the notation $f_\varphi\colon X \to Y, f_\varphi(x) = f(x)(\varphi)$. This simplifies Kleisli composition in the following arguments because

$$(g \circ f)_\varphi = g_\varphi \cdot f_\varphi \qquad \text{for any } f\colon X \mathrel{-\!\!\circ\!\!\to} Y, g\colon Y \mathrel{-\!\!\circ\!\!\to} Z.$$

**Definition 5.13.** For a functor $H\colon \mathsf{Poset} \to \mathsf{Poset}$ with an extension $\hat{H}\colon \mathsf{Kl}(\_^\Phi) \to \mathsf{Kl}(\_^\Phi)$, a *version filter* is a $\Phi$-indexed family of natural transformations $(|^\varphi\colon \hat{H} \mathrel{-\!\!\circ\!\!\to} \hat{H})_{\varphi\in\Phi}$ such that for every $\varphi \in \Phi$ the following restriction holds:

$$\hat{H}X \;\overset{|^\varphi_X}{\underset{|^\varphi_X}{\rightrightarrows}}\; \hat{H}X \xrightarrow{\hat{H}\overline{\mathrm{id}_X \times (\varphi!)}} \hat{H}(X \times \Phi) \quad \text{and} \quad \big((|^\psi_X)_\varphi\colon HX \to HX\big)_{\psi\in\Phi} \text{ jointly monic.} \tag{5.5}$$

Recall that $\overline{\mathrm{id}_X \times \mathrm{id}_\Phi}\colon X \mathrel{-\!\!\circ\!\!\to} X \times \Phi$ is just the curried version of $\mathrm{id}_X \times \mathrm{id}_\Phi$.

However, it should be noted that $|^\varphi$ is *not* required to be monotone in $\varphi$. In the second of our main examples, $|^\varphi$ is neither monotone nor antitone in $\varphi$.

Intuitively, the first part of (5.5) says that each map $|^\varphi_X$ filters those elements from $\hat{H}X$ that are associated with the version $\varphi \in \Phi$. For instance, $C \in \mathcal{P}(X \times \Phi)$ can contain tuples $(x, \psi)$ holding an arbitrary versions $\psi \in \Phi$, but after filtering by $|^\varphi_X\colon \mathcal{P}(X \times \Phi) \mathrel{-\!\!\circ\!\!\to} \mathcal{P}(X \times \Phi)$ only those terms with $\psi = \varphi$ remain. The second part of (5.5) expresses that each set of behaviours $b \in HX$ is fully determined by the restrictions to all possible versions. Here, it is not enough to require that the $(|^\psi_X)_{\psi\in\Phi}$ are jointly monic in $\mathsf{Kl}(\_^\Phi)$, because there are monos $m\colon X \mathrel{-\!\!\circ\!\!\to} Y$ in $\mathsf{Kl}(\_^\Phi)$ and $\varphi \in \Phi$ s.t. $m_\varphi$ is not monic.[1]

**Proposition 5.14.** *For an extension $\hat{H}\colon \mathsf{Kl}(T) \to \mathsf{Kl}(T)$ of $H$, a family of morphisms $\rho_X\colon HX \to THX$ is a natural transformation $\rho\colon \hat{H} \mathrel{-\!\!\circ\!\!\to} \hat{H}$ iff $\rho_X$ is natural in $X$ with:*

$$\begin{array}{ccc} HTX & \xrightarrow{\rho_{TX}} & HTX \\ {\scriptstyle \hat{H}\mathrm{id}_{TX}}\downarrow & & \downarrow{\scriptstyle \hat{H}\mathrm{id}_{TX}} \\ HX & \xrightarrow{\rho_X} & HX \end{array} \quad \text{in } \mathsf{Kl}(T) \tag{5.6}$$

[1]For instance for $\Phi = \{\varphi' \le \varphi\}$, $m\colon 2 \mathrel{-\!\!\circ\!\!\to} 2_\perp$ is monic, where $2 = \{0, 1\}$ is discrete and $2_\perp = \{0, 1, \perp\}$ is 2 with a bottom element; define $m_\varphi = \mathrm{id}_2$, $m_{\varphi'} = \perp!$.

*where* $\mathrm{id}_{TX}$ *is considered as* $\mathrm{id}_{TX} \colon TX \multimap X$.

Note that $\hat{H}\mathrm{id}_{TX} \colon HTX \to THX$ is the corresponding distributive law w.r.t. Theorem 2.10.

*Proof.*

($\Rightarrow$) Assume $\rho_X \colon \hat{H}X \multimap \hat{H}X$ is natural in $X$. Then (5.6) is just the naturality square for $\mathrm{id}_{TX} \colon TX \multimap X$. For any pure $f \colon X \to Y$, the extension $\hat{H}$ ensures $\hat{H}If = IHf$, and so we have

$$THf \cdot \rho_X = IHf \circ \rho_X = \rho_Y \circ IHf = \rho_Y \cdot Hf,$$

i.e. $\rho_X \colon HX \to THX$ is natural in $X$.

($\Leftarrow$) For $f \colon X \multimap Y$, $\hat{H}f$ can be rewritten as

$$\hat{H}f = \hat{H}(\mathrm{id}_{TY} \cdot f) = \hat{H}(\mathrm{id}_{TY} \circ If) = \hat{H}\mathrm{id}_{TY} \circ \hat{H}If = \hat{H}\mathrm{id}_{TY} \circ IHf$$

Using that $\rho_X \colon HX \to THX$ is natural in $X$, we have

$$\rho_Y \circ \hat{H}f = \rho_Y \circ \hat{H}\mathrm{id}_{TY} \circ IHf \overset{(5.6)}{=} \hat{H}\mathrm{id}_{TY} \circ \rho_{TX} \circ IHf = \hat{H}\mathrm{id}_{TY} \circ (\rho_{TX} \cdot Hf)$$
$$\overset{\text{Naturality}}{=} \hat{H}\mathrm{id}_{TY} \circ (THf \cdot \rho_{TX}) = \hat{H}\mathrm{id}_{TY} \circ IHf \circ \rho_X = \hat{H}f \circ \rho_X \qquad \square$$

**Example 5.15.**

(1) For the standard powerset functor $\mathcal{P}(\_)$, define $|^{\varphi}_X \colon \mathcal{P}X \multimap \mathcal{P}X$

$$(|^{\varphi}_X)(C)(\psi) = \begin{cases} C & \text{if } \varphi = \psi \\ \emptyset & \text{otherwise.} \end{cases}$$

This is natural in $X$ because for $f \colon X \to Y^{\Phi}$ we have

$$(\hat{H}f)_\psi \cdot |^{\varphi}_X(C)(\psi) = (\hat{H}f)(C)(\psi) = (|^{\varphi}_X)_\psi \cdot (\hat{H}f)(C)(\psi) \qquad \text{if } \psi = \varphi$$
$$(\hat{H}f)_\psi \cdot |^{\varphi}_X(C)(\psi) = (\hat{H}f)(\emptyset)(\psi) = \emptyset = (|^{\varphi}_X)_\psi \cdot (\hat{H}f)(C)(\psi) \qquad \text{if } \psi \neq \varphi$$

The first axiom of (5.5) evaluated for $\psi \in \Phi$ can be checked by case distinction on $\psi = \varphi$. If $\psi \neq \varphi$, then $(\hat{H}\overline{\mathrm{id}}_{X \times \Phi})_\psi \cdot (|^{\varphi}_X)_\psi$ is constantly $\emptyset$, and so is the other part of the diagram. If $\psi = \varphi$ then the diagram (5.5) commutes by definition of $\hat{H}$:

$$(\hat{H}\overline{\mathrm{id}_X \times \mathrm{id}_\Phi})_\varphi(C) = \bar{p}_X(C)(\varphi) = \{(x, \varphi) \mid x \in C\}$$
$$= H\langle \mathrm{id}_X, \varphi! \rangle(C) = (\hat{H}\overline{\mathrm{id}_X \times \varphi!})_\varphi(C).$$

The family $((|^{\psi}_X)_\varphi)_{\psi \in \Phi}$ is jointly monic, because already $(|^{\varphi}_X)_\varphi$ is monic.

(2) Since the previous filter is defined for an extension for discrete $\Phi$, the previous filter function also is a filter for $\mathcal{DP}$.

(3) For $\mathcal{P}(\_ \times \Phi)$, first define the family of natural transformations $r^{\varphi} \colon \mathcal{P}(\_ \times \Phi) \to \mathcal{P}(\_ \times \Phi)$

$$r^{\varphi}_X(C) = \{(x, \varphi') \in C \mid \varphi' = \varphi\}$$

and then $|^{\varphi} := \nu \cdot r^{\varphi}$. That is, the version filter is pure, just like the distributive law for this functor (Prop. 5.9). For $f \colon X \times \Phi \to Y$ we have

$$r^{\varphi}_Y \cdot \mathcal{P}(\langle f, \pi_2 \rangle)(C) = \{(y, \varphi') \in \mathcal{P}(\langle f, \pi_2 \rangle)(C) \mid \varphi' = \varphi\}$$
$$= \{(f(x, \varphi'), \varphi') \mid (x, \varphi') \in C \wedge \varphi' = \varphi\}$$
$$= \mathcal{P}(\langle f, \pi_2 \rangle)\{(x, \varphi') \in C \mid \varphi' = \varphi\} = \mathcal{P}(\langle f, \pi_2 \rangle) \cdot r^{\varphi}_X(C).$$

So in particular $r^\varphi$ is natural in $X$ and $r^\varphi_X \cdot \mathcal{P}\langle \mathrm{ev}_X, \pi_2\rangle = \mathcal{P}\langle \mathrm{ev}_X, \pi_2\rangle \cdot r^\varphi_{X^\Phi}$. Hence, $|^\varphi \colon \mathcal{P}(\_ \times \Phi) \multimap \mathcal{P}(\_ \times \Phi)$ is natural. For the axioms (5.5), we have for all $h \colon \Phi \to \Phi$

$$\begin{aligned}(\hat{H}\overline{\mathrm{id}_X \times h})_\psi \cdot (|^\varphi_X)_\psi(C) &\overset{(5.3)}{=} H(\mathrm{id}_X \times h) \cdot (\bar{\Lambda}_X)_\psi \cdot r^\varphi_X(C)\\ &\overset{(5.4)}{=} \mathcal{P}(\mathrm{id}_X \times h \times \mathrm{id}_\Phi) \cdot \mathcal{P}(\mathrm{id}_X \times \Delta_\Phi) \cdot r^\varphi_X(C)\\ &= \mathcal{P}(\mathrm{id}_X \times \langle h, \mathrm{id}_\Phi\rangle)(\{(x, \varphi') \in C \mid \varphi' = \varphi\})\\ &= \{(x, h(\varphi'), \varphi') \in C \mid \varphi' = \varphi\}.\end{aligned}$$

Clearly, $\{(x, \varphi!(\varphi'), \varphi') \in C \mid \varphi' = \varphi\} = \{(x, \mathrm{id}_\Phi(\varphi'), \varphi') \in C \mid \varphi' = \varphi\}$, so (5.5) commutes. The family $(r^\psi_X)_{\psi\in\Phi}$ is jointly monic since, for any $t_1, t_2 \in \mathcal{P}(X \times \Phi)$ with $r^\psi_X(t_1) = r^\psi_X(t_2)$, we have

$$t_1 = \bigcup_{\psi\in\Phi} r^\psi_X(t_1) = \bigcup_{\psi\in\Phi} r^\psi_X(t_2) = t_2.$$

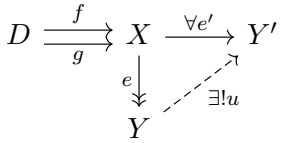
So for all $\varphi \in \Phi$, $(|^\psi_{X\,\varphi})_{\psi\in\Phi}$ is jointly-monic, because $(|^\psi_X)_\varphi = r^\psi_X$.

(4) For $\mathcal{V}X = \mathcal{P}(X \times \Phi)^A$ apply the previous filter component-wise, i.e. $(|^\varphi)^A$. When considering a CTS as a coalgebra for $\mathcal{P}(\_ \times \Phi)$ (5.2), the filter recovers the structure of the underlying transition system for a version $\varphi \in \Phi$ as follows:

$$|^\varphi_X \cdot \alpha_\varphi(x)(a) = \{(x', \varphi) \mid x \xrightarrow{\varphi, a} x'\}.$$

**Definition 5.16** (Coequaliser)**.** Recall that for a parallel pair of morphisms $f, g \colon D \rightrightarrows X$, the coequaliser of $f$ and $g$ is a morphism $e \colon X \to Y$ such that

(1) $e$ merges $f$ and $g$, i.e., $e \cdot f = e \cdot g$.
(2) $e$ is the least such morphism, i.e., for any $e' \colon X \to Y$ with $e' \cdot f = e' \cdot g$, there is a unique $u \colon Y \to Y'$ with $e' = u \cdot e$ as indicated in the following diagram.




A morphism is called a *regular epimorphism* if it is the coequaliser for a pair of morphisms. In Set and in preorders, $Y$ is the quotient of $X$ by the reflexive, symmetric, transitive closure of the relation $\{(f(d), g(d)) \mid d \in D\}$. In Poset, the first step is to construct $Y$ as in preorders, and in a second step, additional elements are identified due to antisymmetry.

**Notation 5.17.** Instead of writing a relation $E \subseteq X \times X$, we consider its projections $\pi_1, \pi_2 \colon E \rightrightarrows X$ as morphisms, usually by writing a relation as $E \rightrightarrows X$. Then the quotient of $X$ by $E$, denoted by ${}^X\!/_E$ is the coequaliser of the projections $E \rightrightarrows X$. If $E$ is already an equivalence relation, then this is the usual quotient (with additionally identified elements due to antisymmetry in Poset).

Next, we lift the notion of conditional bisimulation to the level of coalgebras over the base category Poset. As a result, one can reason with conditional bisimilarity for any systems whose behavioural functor $\hat{H} \colon \mathsf{Kl}(\_^\Phi) \to \mathsf{Kl}(\_^\Phi)$ comes with a notion of version filter $|^\varphi$.

**Definition 5.18.** Given a coalgebra $\alpha \colon X \to HX^\Phi$ for a functor with a version filter.

(1) a *conditional bisimulation* for $\alpha$ is a $\Phi$-indexed family of relations $R_\varphi \rightrightarrows X$ such that

(a) $\varphi' \leq \varphi$ implies $R_{\varphi'} \supseteq R_\varphi$.
(b) $R_\varphi$ is a bisimulation for the $H$-coalgebra $(|_X^\varphi)_\varphi \cdot \alpha_\varphi$.

(2) a *conditional congruence* is a $\Phi$-indexed family of relations $R_\varphi \rightrightarrows X$ such that
(a) $\varphi' \leq \varphi$ implies $R_{\varphi'} \supseteq R_\varphi$.
(b) the projections $R_\varphi \rightrightarrows X$ are made equal by the morphism:

$$X \xrightarrow{\alpha_\varphi} HX \xrightarrow{(|_X^\varphi)_\varphi} HX \xrightarrow{H\kappa_\varphi} H^X\!/_{R_\varphi},$$

where $\kappa_\varphi$ is the coequaliser of the parallel arrows $R_\varphi \rightrightarrows X$.

Just like in the traditional coalgebraic setup, the notions conditional congruence and conditional bisimulation coincide if the underlying endofunctor $H$ preserves weak pullbacks. This is the case for both $\mathcal{P}(\_)^A$ and $\mathcal{P}(\_ \times \Phi)^A$. Furthermore, it should be noted that the above notion of conditional bisimilarity for both the cases $\mathcal{P}(\_)^A$ and $\mathcal{P}(\_ \times \Phi)^A$ coincides with the concrete definition conditional bisimilarity (cf. Definition 3.2).

**Definition 5.19.** We say that a $\hat{H}$-coalgebra $\alpha\colon X \mathrel{-\!\!\circ\!\!\to} \hat{H}X$ *preserves upgrades* if

$$(|_X^\psi \circ \alpha)_\varphi = (|_X^\psi \circ \alpha)_\psi \qquad \text{for all } \psi \leq \varphi \text{ in } \Phi \tag{5.7}$$

$$(|_X^\psi \circ \alpha)_\varphi \text{ is constant} \qquad \text{for all } \psi \not\leq \varphi \text{ in } \Phi. \tag{5.8}$$

Intuitively, (5.7) says that a state always has the same $\psi$ successors in a version $\psi$, no matter whether the state is already in the version $\psi$ or can upgrade to the version $\psi$. The second property in (5.8) asserts that the successors of a state in two different version $\psi, \psi'$ (which cannot be upgraded from $\varphi$) remain the same. In our working examples, we have $(|_X^\psi \circ \alpha)_\varphi = \emptyset = (|_X^{\psi'} \circ \alpha)_\varphi$ for any $\psi, \psi' \not\leq \varphi$.

**Example 5.20.** The coalgebras modelling CTS in Remark 5.3 indeed preserve upgrades:

(1) For $\mathcal{P}(\_)^A$ and discrete $\Phi$, all coalgebras satisfy the conditions: $\psi \leq \varphi$ in (5.7) boils down to $\psi = \varphi$ and (5.7) becomes trivial; similarly, $\psi \not\leq \varphi$ boils down to $\psi \neq \varphi$, and indeed $(|_X^\psi \circ \alpha)_\varphi = (|_X^\psi)_\varphi \cdot \alpha_\varphi$ is constantly $\emptyset$ by the definition of $|^\psi$.
(2) For the coalgebra modelling a CTS with upgrades

$$\alpha(x)(\varphi)(a) = \{(x', \varphi') \mid x \xrightarrow{a,\varphi'} x' \wedge \varphi' \leq \varphi\}. \qquad \text{(cf. 5.2)}$$

for any $\psi, \varphi \in \Phi$, we have

$$(|_X^\psi)_\varphi \cdot \alpha_\varphi(x)(a) = \{(x', \psi) \mid x \xrightarrow{\psi,a} x'\} = (|_X^\psi)_\psi \cdot \alpha_\psi(x)(a) \qquad \text{if } \psi \leq \varphi$$

$$(|_X^\psi)_\varphi \cdot \alpha_\varphi(x)(a) = \emptyset \qquad \text{if } \psi \not\leq \varphi.$$

For upgrade preserving coalgebras, we can show that the concrete notion of behavioural equivalence coincides with the coalgebraic notion. We first need the following lemma.

**Lemma 5.21.** *Given a functor $H$ with a version filter and a coalgebra $\alpha\colon X \mathrel{-\!\!\circ\!\!\to} \hat{H}X$ which preserves upgrades, then for any $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$, $\varphi \in \Phi$, and $x_1, x_2 \in X$, we have*

$$x_1, x_2 \text{ are merged by } X \xrightarrow{\alpha_\varphi} HX \xrightarrow{(\hat{H}f)_\varphi} HY$$

$$\iff$$

$$x_1, x_2 \text{ are merged by } X \xrightarrow{\alpha_\psi} HX \xrightarrow{(|_X^\psi)_\psi} HX \xrightarrow{Hf_\psi} HY \quad \text{for all } \psi \leq \varphi.$$

*Proof.* Since the $(|_Y^\psi)_\varphi$, $\psi \in \Phi$, are jointly-monic we have:

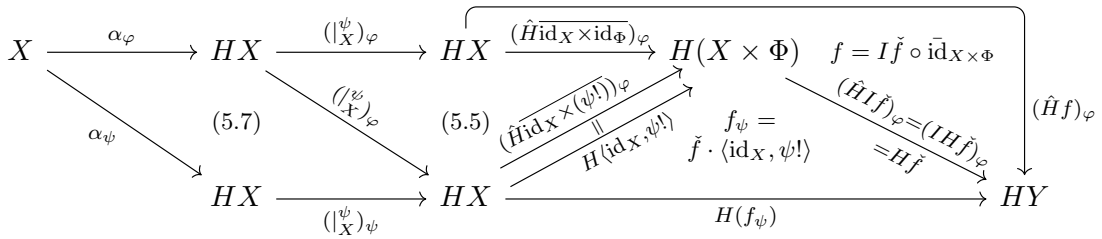



Figure 1: Commutative diagram showing the connection between $\alpha_\varphi$ and $\alpha_\psi$, $\psi \leq \varphi$, when uncurrying $f$ to $\check{f}\colon X \times \Phi \to Y$

$x_1, x_2$ are merged by $(\hat{H}f)_\varphi \cdot \alpha_\varphi$

$\Leftrightarrow$ for all $\psi \in \Phi$, $x_1, x_2$ are merged by $(|^\psi_Y)_\varphi \cdot (\hat{H}f)_\varphi \cdot \alpha_\varphi = (|^\psi_Y \circ \hat{H}f \circ \alpha)_\varphi$

By the naturality of $|^\psi\colon \hat{H} \multimap \hat{H}$ we have:

$\Leftrightarrow$ for all $\psi \in \Phi$, $x_1, x_2$ are merged by $(\hat{H}f)_\varphi \cdot (|^\psi_X)_\varphi \cdot \alpha_\varphi = (\hat{H}f \circ |^\psi_X \circ \alpha)_\varphi$

$\Leftrightarrow$ for all $\psi \leq \varphi$, $x_1, x_2$ are merged by $(\hat{H}f)_\varphi \cdot (|^\psi_X)_\varphi \cdot \alpha_\varphi$ and
for all $\psi \not\leq \varphi$, $x_1, x_2$ are merged by $(\hat{H}f)_\varphi \cdot (|^\psi_X)_\varphi \cdot \alpha_\varphi$

By (5.8), $(|^\psi_X)_\varphi \cdot \alpha_\varphi$ is constant for $\psi \not\leq \varphi$ and thus the second conjunct is vacuous.

$\Leftrightarrow$ for all $\psi \leq \varphi$, $x_1, x_2$ are merged by $(\hat{H}f)_\varphi \cdot (|^\psi_X)_\varphi \cdot \alpha_\varphi$

By the commutativity of Figure 1, we finally have the desired equivalence:

$\Leftrightarrow$ for all $\psi \leq \varphi$, $x_1, x_2$ are merged by $H(f_\psi) \cdot (|^\psi_X)_\psi \cdot \alpha_\psi$ □

The above lemma highlights an important property of upgrade preserving coalgebra; namely that two states $x_1, x_2$ have the same set of successors for a condition $\varphi$ under the image of $\hat{H}(f)$ (where $f$ is an arrow in $\mathsf{Kl}(\_^\Phi)$) if and only if they have the same set of successors for every upgrade $\psi \leq \varphi$. With this we can finally prove the main statement:

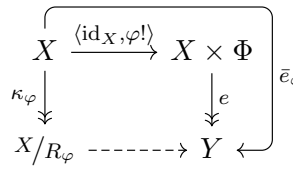
**Theorem 5.22.** *Let $H\colon \mathsf{Poset} \to \mathsf{Poset}$ preserve monos and have a version filter. Then for an upgrade preserving $\hat{H}$-coalgebra $\alpha\colon X \multimap \hat{H}X$, states $x_1, x_2 \in X$ are conditionally congruent in $\varphi$ iff there is a $\hat{H}$-coalgebra homomorphism $h$ with $h(x_1)(\varphi) = h(x_2)(\varphi)$.*

*Proof.*

($\Rightarrow$) Given a conditional behavioural equivalence $(R_\varphi)_{\varphi\in\Phi}$, define $E \rightrightarrows X \times \Phi$ as the relation

$$E := \{((x_1, \varphi), (x_2, \varphi)) \mid (x_1, x_2) \in R_\varphi\}$$

and let $e\colon X \times \Phi \to Y$ be the coequaliser of the projections of $E$. By definition, $e(x_1, \varphi) = e(x_2, \varphi)$ for all $(x_1, x_2) \in R_\varphi$. So diagramatically speaking, the coequaliser ${}^X\!/_{R_\varphi}$ induces a unique morphism with




for all $\varphi \in \Phi$. (5.9)

It remains to show that $\bar{e}\colon X \mathrel{-\!\!\circ\!\!\to} Y$ is the carrier of some $\hat{H}$-coalgebra homomorphism. The necessary coalgebra structure on $Y$ will be induced by the coequaliser $e$. So fix $((x_1,\varphi),(x_2,\varphi)) \in E$, hence $x_1, x_2 \in R_\varphi$ and we have:

$$\begin{aligned}
& x_1, x_2 \in R_\psi \text{ for all } \psi \leq \varphi \\
\overset{\text{Def. 5.18}}{\Longrightarrow}\ & x_1, x_2 \text{ are merged by } H\kappa_\psi \cdot (|^\psi_X)_\psi \cdot \alpha_\psi \text{ for all } \psi \leq \varphi \\
\overset{(5.9)}{\Longrightarrow}\ & x_1, x_2 \text{ are merged by } H\bar{e}_\psi \cdot (|^\psi_X)_\psi \cdot \alpha_\psi \text{ for all } \psi \leq \varphi \\
\overset{\text{Lem. 5.21}}{\Longrightarrow}\ & x_1, x_2 \text{ are merged by } (\hat{H}\bar{e})_\varphi \cdot \alpha_\varphi
\end{aligned}$$

So, the projections $E \rightrightarrows X \times \Phi$ are merged by the uncurried version of the morphism $\hat{H}\bar{e} \circ \alpha\colon X \mathrel{-\!\!\circ\!\!\to} HY$, i.e., by $u(x,\varphi) := (\hat{H}\bar{e} \circ \alpha)_\varphi(x)$. Hence, the coequaliser $e$ induces a unique morphism $\beta\colon Y \to HY$ with:

$$\begin{array}{ccccc}
E \rightrightarrows X \times \Phi & & X & & X \xrightarrow{\alpha} HX \\
{\scriptstyle e}\downarrow \quad \searrow^{u} & \Longrightarrow & {\scriptstyle \bar{e}}\downarrow \quad \searrow^{\hat{H}\bar{e}\circ\alpha} & \Longrightarrow & {\scriptstyle \bar{e}}\downarrow \qquad \downarrow{\scriptstyle \hat{H}\bar{e}} \\
Y \xrightarrow[\beta]{} HY & & Y^\Phi \xrightarrow[\beta^\Phi]{} HY^\Phi & & Y \xrightarrow[I\beta]{} HY
\end{array}$$

($\Leftarrow$) We prove directly that the family

$$R_\varphi := \{(x_1,x_2) \in X \times X \mid \text{there is some } \hat{H}\text{-coalgebra } h \text{ with } h_\varphi(x_1) = h_\varphi(x_2)\}$$

is a conditional congruence:

(a) Let $\varphi' \leq \varphi$ and let $(x_1,x_2) \in R_\varphi$ be witnessed by $h\colon (X,\alpha) \mathrel{-\!\!\circ\!\!\to} (Y,\beta)$. Then $x_1, x_2$ are merged by $\beta_\varphi \cdot h_\varphi = (\hat{H}h)_\varphi \cdot \alpha_\varphi$. Applying Lemma 5.21 first forward, restricting to $\varphi' \leq \varphi$ and then applying Lemma 5.21 backwards again, shows that $x_1, x_2$ are merged by $(\hat{H}h)_{\varphi'} \cdot \alpha_{\varphi'}$. Since $\hat{H}h \circ \alpha = \beta \circ h$ is the composition of the $\hat{H}$-coalgebra homomorphisms $h$ and $\beta$, we have the witness for $(x_1,x_2) \in R_{\varphi'}$.

(b) Let $(x_1,x_2) \in R_\varphi$, witnessed by $h\colon (X,\alpha) \mathrel{-\!\!\circ\!\!\to} (Y,\beta)$. So we have $(p_1,p_2) \in R_\varphi$ for all $p_1, p_2$ with $h_\varphi(p_1) = h_\varphi(p_2)$. Then the regular epi part $e$ of $h_\varphi$ induces a unique $u$ such that:

$$\begin{array}{ccccc}
X & \xrightarrow{e} & K & \xrightarrowtail{m} & Y \\
& {\scriptstyle \kappa_\varphi}\searrow & {\scriptstyle u}\downarrow & & \\
& & X/_{R_\varphi} & &
\end{array}$$

Denote the mono part of $h_\varphi$ by $m$, and so $Hm$ is monic too. Finally:

$$\begin{aligned}
& (x_1,x_2) \in R_\varphi \\
\Longrightarrow\ & x_1, x_2 \text{ are merged by } \beta_\varphi \cdot h_\varphi = (\beta \circ h)_\varphi = (\hat{H}h \circ \alpha)_\varphi \\
\overset{\text{Lem. 5.21}}{\Longrightarrow}\ & x_1, x_2 \text{ are merged by } Hh_\psi \cdot (|^\psi_X)_\psi \cdot \alpha_\psi \text{ for all } \psi \leq \varphi \\
\overset{Hm \text{ monic}}{\Longrightarrow}\ & x_1, x_2 \text{ are merged by } He \cdot (|^\psi_X)_\psi \cdot \alpha_\psi \text{ for all } \psi \leq \varphi \\
\overset{\kappa_\varphi = u\cdot e}{\Longrightarrow}\ & x_1, x_2 \text{ are merged by } H\kappa_\varphi \cdot (|^\psi_X)_\psi \cdot \alpha_\psi \text{ for all } \psi \leq \varphi \\
\Longrightarrow\ & x_1, x_2 \text{ are merged by } H\kappa_\varphi \cdot (|^\varphi_X)_\varphi \cdot \alpha_\varphi.
\end{aligned}$$

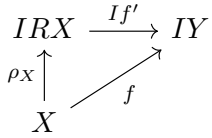
□

Note that both $\mathcal{P}$ and $\mathcal{P}(\_ \times \Phi)$ preserve monos, i.e. monotone maps with injective carrier, in $\mathsf{Poset}$. Hence, Theorem 5.22 holds for both functors and so coalgebraic behavioural equivalence coincides with conditional bisimilarity.

## 6.Computing Behavioural Equivalence

In this section, we concentrate on algorithms to obtain a minimal CTS from a given CTS up to conditional bisimilarity. Therefore, the final chain algorithm for minimisation from [ABH$^+$12] is applied to the CTS functors $\mathcal{DP}$ and $\mathcal{P}(\_ \times \Phi)$. The algorithm performs minimisation and determinisation for coalgebras on a Kleisli category, in which the pure arrows form a reflective subcategory:

**Definition 6.1.** A subcategory $\mathcal{S}$ of $\mathcal{A}$ is called *reflective*, if the inclusion functor $I\colon \mathcal{S} \hookrightarrow \mathcal{A}$ has a left-adjoint $R$. The spelled out adjunction means: For each $X \in \mathcal{A}$ there is an $\mathcal{S}$-object $RX$ and an $\mathcal{A}$-arrow $\rho_X\colon X \to IRX$ such that for any $\mathcal{A}$-arrow $f\colon X \to IY$ into some object $Y$ of $\mathcal{S}$, there exists a *unique* $\mathcal{S}$-arrow $f'\colon RX \to Y$ (called $\rho$-reflection of $f$) such that:




Note that for such a mapping $R\colon \mathbf{obj}\,\mathcal{A} \to \mathbf{obj}\,\mathcal{S}$ on objects, $R$ uniquely extends to a functor $R\colon \mathcal{A} \to \mathcal{S}$, and is called *reflector*.

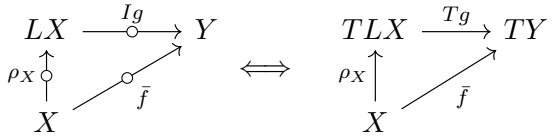
**Remark 6.2.** Here, the definition of [AHS90] is followed and thus the subcategory $\mathcal{S} \hookrightarrow \mathcal{A}$ is not required to be full. This is important because the pure arrows need to form a reflective subcategory of the Kleisli category, i.e., we have a reflective subcategory $\mathcal{C} \hookrightarrow \mathsf{Kl}(T)$ for a monad $T\colon \mathcal{C} \to \mathcal{C}$. And this subcategory is full if and only if $T$ is the identity monad.

For the reader monad $\_^{\Phi}$ on $\mathsf{Poset}$, we have a non-full reflective subcategory:

**Lemma 6.3.** *For a monad $(T\colon \mathcal{C} \to \mathcal{C}, \eta, \mu)$, the base category $\mathcal{C} \hookrightarrow \mathsf{Kl}(T)$ is a reflective subcategory iff $T$ has a left-adjoint $L\colon \mathcal{C} \to \mathcal{C}$. Furthermore, the unit $\rho_X\colon X \mathrel{-\!\!\circ\!\!\to} LX$ of the adjunction $L \dashv T$ is the universal arrow of the reflection.*

*Proof.*

($\Leftarrow$) For $L \dashv T$ with unit $\rho$, for all $\bar{f}\colon X \mathrel{-\!\!\circ\!\!\to} Y$, and $g\colon LX \to Y$, we have $Ig \circ \rho_X = Tg \cdot \rho_X$, and so




The first diagram is the universal property of the reflection, the last is that of $L \dashv T$; so the direction from right to left proves existence of an reflection of $\bar{f}$ and the direction from left to right proves its uniqueness.

($\Rightarrow$) Let $R \dashv I$ and note that with the forgetful $U\colon \mathsf{Kl}(T) \to \mathcal{C}$ from the Kleisli adjunction $I \dashv U$, we have $T = UI$. Define $L := RI$; then $L \dashv T$ by following natural isomophisms between hom-sets:

$$\dfrac{\dfrac{RI\,X \longrightarrow Y \text{ in } \mathcal{C}}{IX \mathrel{-\!\!\circ\!\!\to} IY \text{ in } \mathsf{Kl}(T)}\; R \dashv I}{X \longrightarrow UI\,Y \text{ in } \mathcal{C}}\; I \dashv U$$

□

Note that for $R \dashv I$ and $L \dashv T$, $IR\colon \mathsf{Kl}(T) \to \mathsf{Kl}(T)$ is an extension of $L$, because $IL = IRI$.

**Example 6.4.** In case of the reader monad $\_^{\Phi}$ on $\mathcal{C} = \mathsf{Poset}$ or $\mathcal{C} = \mathsf{Set}$ (resp. $\Phi := \mathcal{J}(\mathbb{L})$, and $\_^{\mathcal{J}(\mathbb{L})} \cong T$ the lattice monad from Section 4), consider a Kleisli arrow $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$. Then its reflection is the uncurried $\check{f}$:

$$\check{f}\colon \overbrace{X \times \Phi}^{LX} \to Y \quad \check{f}(x,\varphi) = f(x)(\varphi).$$

The reflector $R\colon \mathsf{Kl}(\_^{\Phi}) \to \mathcal{C}$ maps $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$ to the pure map

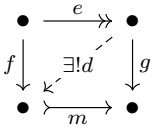
$$Rf\colon X \times \Phi \longrightarrow Y \times \Phi, \quad Rf(x,\varphi) = (f(x)(\varphi), \varphi)$$

which is the reflection of $\rho_Y \circ f\colon X \mathrel{-\!\!\circ\!\!\to} Y \times \Phi$.

Following [ABH+12], a reflective subcategory with an $(\mathcal{E},\mathcal{M})$-factorisation structure gives rise to a pseudo-factorisation structure in the base category, which in turn can be used to compute behavioural equivalence, provided the functor meets some conditions.

**Definition 6.5.** Let $\mathcal{E}$ and $\mathcal{M}$ be any two classes of morphisms in a category $\mathcal{S}$. Then the tuple $(\mathcal{E},\mathcal{M})$ is called a *factorisation structure* for $\mathcal{S}$ if

- The classes $\mathcal{E}$ and $\mathcal{M}$ are closed under composition with isomorphisms;
- Every arrow $f$ of $\mathcal{S}$ has a factorisation $f = m \cdot e$, where $m \in \mathcal{M}$ and $e \in \mathcal{E}$;
- *Unique diagonal property*: For all arrows $f, g$, $e \in \mathcal{E}$, and $m \in \mathcal{M}$, if $g \cdot e = m \cdot f$, then there exists a unique arrow $d$ such that:




(6.1)

**Remark 6.6.** In case of $\mathcal{E}$ = regular epimorphisms and $\mathcal{M}$ = monomorphisms the diagonalisation property (6.1) holds automatically. If $e$ is the coequaliser of $p_1, p_2$, then $e$ merges $p_1$ and $p_2$ and so does $g \cdot e = m \cdot f$. Since $m$ is monic, $f \cdot p_1 = f \cdot p_2$ and the coequaliser $e$ induces a unique diagonal with $d \cdot e = f$ and thus also $m \cdot d = g$.

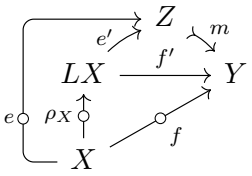
**Example 6.7.** Poset has a (RegEpi,Mono)-factorisation structure [AHS90, 14.23 Examples].

- Regular epimorphisms in Poset are monotone functions $e\colon X \to Y$ where $e$ is surjective and $\leq_Y$ is the smallest order on $Y$ making $e$ monotone. Regular epimorphisms are by definition coequalisers. In complete categories such as Poset, a regular epimorphism is the coequaliser of its kernel pair $\ker e \rightrightarrows X$.
- Monos in Poset are monotone maps with an injective carrier map. In other words, $U\colon \mathsf{Poset} \to \mathsf{Set}$ creates monos.

One transfers the factorisation structure from Poset to $\mathsf{Kl}(\_^{\Phi})$ using the reflection:

**Notation 6.8.** From now on, the application of the inclusion functor $\mathcal{S} \hookrightarrow \mathcal{A}$ is made implicit. For clarity, morphisms in $\mathcal{S}$ are denoted by $\longrightarrow$, and morphisms in $\mathcal{A}$ by $\mathrel{-\!\!\circ\!\!\to}$. $\mathcal{S}$-arrows in $\mathcal{M}$ are indicated by $\rightarrowtail$.

**Definition 6.9.** Consider a reflective subcategory $\mathcal{S} \hookrightarrow \mathcal{A}$ with the $\mathcal{A}$-reflection $\rho$ and with an $(\mathcal{E},\mathcal{M})$-factorisation structure on $\mathcal{S}$. For an $\mathcal{A}$-morphism $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$, take its reflection $f'\colon LX \to Y$ and construct its $(\mathcal{E},\mathcal{M})$-factorisation $f' = m \cdot e'$ with $e' \in \mathcal{E}$, $m \in \mathcal{M}$. Then $(m, e' \circ \rho_X)$ is called the $(\mathcal{E},\mathcal{M})$-*pseudo factorisation* of $f$.



For such pseudo-factorisations, we do not necessarily have a diagonal arrow for $m \in \mathcal{M}$, $e' \in \mathcal{E}$, and $e = e' \circ \rho_X$ in (6.1), but one can show that such an arrow exists whenever $g$ is in $\mathcal{S}$. And the diagonal will also be an $\mathcal{S}$-arrow.

**Remark 6.10.** This applies to $\mathsf{Poset} \hookrightarrow \mathsf{Kl}(\_^{\Phi})$. The pseudo-factorisation of $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$ in $\mathsf{Kl}(\_^{\Phi})$ is as follows:

- The inclusion $m\colon Y_0 \hookrightarrow Y$, $Y_0 = \{f(x)(\varphi) \mid x \in X, \varphi \in \Phi\} \subseteq Y$.
- The function $e\colon X \to {Y_0}^{\Phi}$ defined as $e(x)(\varphi) = f(x)(\varphi)$.
- The relation $\leq_{Y_0^{\Phi}}$ is the smallest order such that $e'$ is order preserving.

We now recall the algorithm from [ABH+12] in its entirety.

**Algorithm 6.11.** Let $\mathcal{A}$ be a category with a final object 1 and let $\mathcal{S}$ be a complete and reflective subcategory of $\mathcal{A}$ that has an $(\mathcal{E}, \mathcal{M})$-factorisation structure where

- all arrows in $\mathcal{E}$ are epimorphism and
- for all objects $X$ the class of $\mathcal{E}$ morphisms with domain $X$ is a set.

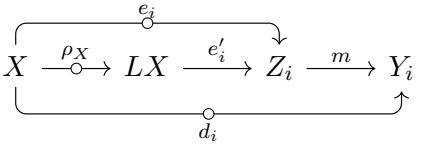
Furthermore, let $\hat{H}$ be an endofunctor on $\mathcal{A}$ preserving $\mathcal{S}$ and $\mathcal{M}$. Then, given an $\hat{H}$-coalgebra $\alpha\colon X \mathrel{-\!\!\circ\!\!\to} \hat{H}X$ we can compute the minimisation of $\alpha$ in the following way:

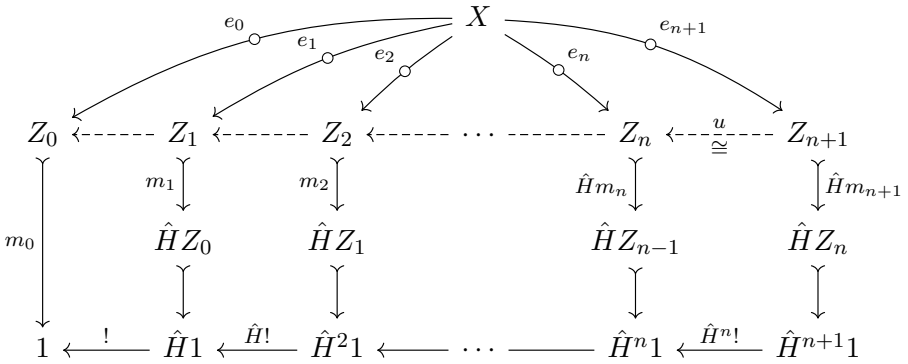
(1) Let $d_0\colon X \mathrel{-\!\!\circ\!\!\to} 1$ be the final morphism.

(2) Given a $d_i\colon X \mathrel{-\!\!\circ\!\!\to} Y$, pseudo-factorise $d_i = m_i \circ e_i$, where $m_i \in \mathcal{M}$, $e_i = e_i' \circ \rho_X$, $e_i' \in \mathcal{E}$.

(3) Compute $d_{i+1} = \hat{H}e_i \circ \alpha$.

(4) The algorithm terminates if the diagonal $u$ with $e_n = u \circ e_{n+1}$ is an isomorphism in $\mathcal{S}$ and yields $e_n$ as its output.



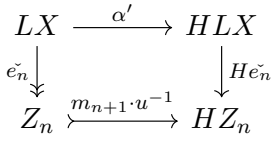

The dashed arrows in the diagram above are obtained by diagonalisation.

Termination is guaranteed whenever the state set $X$ is finite. Whenever the algorithm terminates we obtain a coalgebra homomorphism $e_n$ from $\alpha$ to $m_{n+1} \cdot u^{-1}\colon Z_n \to \hat{H}Z_n$.

The Algorithm 6.11 is correct in the following sense:

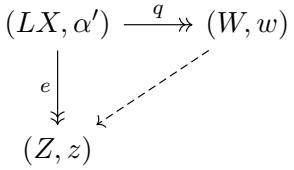
**Theorem 6.12** [ABH+12, Theorem 4.9, Theorem 3.8]**.**
*Let $\alpha'\colon LX \to HLX$ be the reflection of*

$$X \mathrel{\overset{\alpha}{-\!\!\circ\!\!\to}} HX \mathrel{\overset{H\rho_X}{-\!\!\circ\!\!\to}} HLX,$$

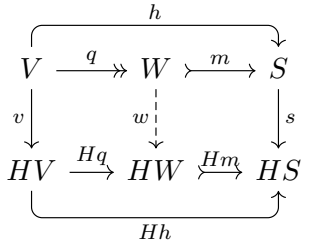



*then the uncurrying of $e_n$, $e_n^{\check{}}\colon LX \to Z_n$, is the greatest $\mathcal{E}$-quotient of $\alpha'$.*

**Remark 6.13.** We call $e\colon (LX,\alpha') \twoheadrightarrow (Z,z)$ the greatest $\mathcal{E}$-quotient if $e \in \mathcal{E}$ and for any $H$-coalgebra homomorphism $q\colon (LX,\alpha') \twoheadrightarrow (W,w)$ with $q \in \mathcal{E}$, there is a unique homomorphism $(W,w) \to (Z,z)$. In $\mathsf{Poset}$ this means that any two elements $x_1, x_2 \in LX$ are merged by $e$ if and only if they are merged by a coalgebra homomorphism in $\mathcal{E}$.




**Remark 6.14.** If $H$ preserves $\mathcal{M}$, then the $(\mathcal{E},\mathcal{M})$-factorisation system lifts to coalgebras, i.e. any coalgebra homomorphism $h$ factorises into $h = m \cdot q$ where $m \in \mathcal{M}$ and $q \in E$ are coalgebra homomorphisms. By the diagonalisation, the coalgebra structure on the image is defined uniquely. So in $\mathsf{Poset}$ for monos $\mathcal{M}$, $x, y \in V$ are merged by some coalgebra homomorphism if and only if they are merged by some $\mathcal{E}$-carried coalgebra homomorphism.




The algorithm's output $e_n\colon X \multimap Z_n$ characterises conditional bisimilarity (in the general sense of Definition 5.18) in the following way: for two elements $x_1, x_2 \in X$ and $\varphi \in \Phi$ we have $x_1 \sim_\varphi x_2$ if and only if $(x,\varphi)$ and $(y,\varphi)$ are merged by the uncurried $\check{e}_n\colon LX \to Z_n$. This characterisation is sound and complete whenever the endofunctor $\hat{H}\colon \mathsf{Kl}(\_^\Phi) \to \mathsf{Kl}(\_^\Phi)$ preserves the subcategory $\mathsf{Poset}$ and the class $\mathcal{M}$:

**Theorem 6.15.** *Using the terminology of Algorithm 6.11 it holds that for $x_1, x_2 \in X$ $x_1 \sim_\varphi x_2$ iff $\check{e}_n$ merges $(x_1,\varphi),(x_2,\varphi)$.*

*Proof.*

($\Leftarrow$) By Theorem 5.22 we know that $x_1 \sim_\varphi x_2$ iff there exists a coalgebra homomorphism $h\colon X \multimap Y$ with $h(x_1)(\varphi) = h(x_2)(\varphi)$. Hence if $\check{e}_n$ merges $(x_1,\varphi),(x_2,\varphi)$, we can infer that $e_n(x_1)(\varphi) = e_n(x_2)(\varphi)$ and since $e_n$ is a coalgebra homomorphism we have $x_1 \sim_\varphi x_2$.

($\Rightarrow$) By Theorem 5.22, $x_1 \sim_\varphi x_2$ implies the existence of some $h\colon (X,\alpha) \multimap (Y,\beta)$ with $h(x_1)(\varphi) = h(x_2)(\varphi)$. Recall from [ABH+12, Prop. 4.4] that since $\hat{H}$ is an extension and $\mathsf{Kl}(\_^\Phi) \hookrightarrow \mathsf{Poset}$ is a reflective subcategory, the category of $H$-coalgebras is a reflective subcategory of the $\hat{H}$-coalgebras. So applying the reflector $R\colon \mathsf{Kl}(\_^T) \to \mathsf{Poset}$ to the square of the $\hat{H}$-coalgebra homomorphism $h$ results in an $H$-coalgebra homomorphism:

$$\begin{array}{ccc} LX & \xrightarrow{\alpha'} & HLX \\ {\scriptstyle Rh}\downarrow & & \downarrow{\scriptstyle HRh} \\ LY & \xrightarrow{\beta'} & HLY \end{array} \qquad \text{in } \mathsf{Poset}$$

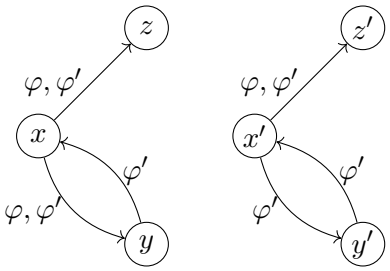
$R$ acts on objects as $L$ and we have $Rh(x_i,\varphi) = (h(x_i)(\varphi),\varphi)$, for both $i \in \{1,2\}$ (cf. Example 6.4), and so $Rh(x_1,\varphi) = Rh(x_2,\varphi)$. By Remark 6.14 and 6.13, the greatest $\mathcal{E}$-quotient $\check{e}_n\colon LX \to Z_n$ merges $(x_1,\varphi)$ and $(x_2,\varphi)$. □

Recall from Theorem 2.10, that the functors $\widehat{\mathcal{D}\mathcal{P}}$ and $\widehat{\mathcal{P}(\_ \times \Phi)}$ (cf. Remark 5.3) preserve the subcategory $\mathsf{Poset}$. Furthermore, they preserve $\mathcal{M}$, i.e., the class of (pure) order preserving injections, because the underlying endofunctors $\mathcal{D}$, $\mathcal{P}$ and $\_ \times \Phi$ do.

Thus, the algorithm from [ABH+12] is applicable using the derived pseudo-factorisation structure. We now discuss a small example for the application of the minimisation algorithm from [ABH+12] using this pseudo-factorisation structure on $\mathsf{Kl}(\_^\Phi)$ for $\mathcal{P}(\_ \times \Phi)$.

**Example 6.16.** Let $X = \{x, y, z, x', y', z'\}$, $|A| = 1$ and $\Phi = \{\varphi', \varphi\}$, with $\varphi' \leq_\Phi \varphi$. Let $\alpha\colon X \to \widehat{\mathcal{V}}X$ (note that $\mathcal{V} = \mathcal{P}(\_ \times \Phi)^A$) be the coalgebra modelling the CTS depicted below.

$\alpha$:




To compute behavioural equivalence, we start by taking the unique morphism $d_0\colon X \to 1$ into the final object of $\mathsf{Kl}(\_^\Phi)$ that is $1 = \{\bullet\}$. At the $i^{\text{th}}$ iteration, we obtain $e_i$ via the pseudo-factorisation of $d_i = m_i \circ e_i$ and then we build $d_{i+1} = \widehat{\mathcal{V}}e_i \circ \alpha$. These iterations are shown in the following tables. Note that each table represents both, $d_i$ and $e_i$, because the pseudo-factorisation just yields simple injections as monomorphisms, so $d_i$ and $e_i$ in each step only differ by their codomain.

| $d_0, e_0$ | $x$ | $y$ | $z$ | $x'$ | $y'$ | $z'$ |
|---|---|---|---|---|---|---|
| $\varphi$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ |
| $\varphi'$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ |

| $d_1, e_1$ | $x$ | $y$ | $z$ | $x'$ | $y'$ | $z'$ |
|---|---|---|---|---|---|---|
| $\varphi$ | $\{(\bullet, \varphi), (\bullet, \varphi')\}$ | $\{(\bullet, \varphi')\}$ | $\emptyset$ | $\{(\bullet, \varphi), (\bullet, \varphi')\}$ | $\{(\bullet, \varphi')\}$ | $\emptyset$ |
| $\varphi'$ | $\{(\bullet, \varphi')\}$ | $\{(\bullet, \varphi')\}$ | $\emptyset$ | $\{(\bullet, \varphi')\}$ | $\{(\bullet, \varphi')\}$ | $\emptyset$ |

| $d_2, e_2$ | $x$ | $y$ | $z$ | $x'$ | $y'$ | $z'$ |
|---|---|---|---|---|---|---|
| $\varphi$ | ⑤ | ③ | ④ | ① | ③ | ④ |
| $\varphi'$ | ② | ③ | ④ | ② | ③ | ④ |

| $d_3, e_3$ | $x$ | $y$ | $z$ | $x'$ | $y'$ | $z'$ |
|---|---|---|---|---|---|---|
| $\varphi$ | ⑤ | ③ | ④ | ① | ③ | ④ |
| $\varphi'$ | ② | ③ | ④ | ② | ③ | ④ |

In the tables for $d_2/e_2$ and $d_3/e_3$ we have used colours to code the entries, because the full notation for the entries would be too large to fit in the tables.

The codomains $C_0, C_1, C_2$, and $C_3$ of $e_0, e_1, e_2$, and $e_3$ (resp.) are given below (note that the colours in $C_2$ and $C_3$ indicate the colours in the tables above):

$$C_0 = \{\bullet\}$$

$$C_1 = \{\emptyset, \{(\bullet, \varphi')\}, \{(\bullet, \varphi), (\bullet, \varphi')\}\}$$

$$C_2 = \{\underbrace{\{(\emptyset, \varphi), (\emptyset, \varphi'), (\{(\bullet, \varphi')\}, \varphi')\}}_{①}, \underbrace{\{\{(\emptyset, \varphi'), (\{(\bullet, \varphi')\}, \varphi')\}\}}_{②}, \underbrace{\{(\{(\bullet, \varphi')\}, \varphi')\}}_{③}, \underbrace{\emptyset}_{④},$$
$$\underbrace{\{(\{(\bullet, \varphi')\}, \varphi), (\{(\bullet, \varphi')\}, \varphi'), (\emptyset, \varphi), (\emptyset, \varphi')\}}_{⑤}\}$$

$$C_3 = \{\underbrace{\{\{(\emptyset, \varphi), (\{(\{(\bullet, \varphi')\}, \varphi')\}, \varphi'), (\emptyset, \varphi')\}\}}_{①}, \underbrace{\{\{(\emptyset, \varphi'), (\{(\{(\bullet, \varphi')\}, \varphi')\}, \varphi')\}\}}_{②},$$
$$\underbrace{\{(\{(\{(\bullet, \varphi')\}, \varphi'), (\emptyset, \varphi')\}, \varphi')\}}_{③}, \underbrace{\emptyset}_{④}$$
$$\underbrace{\{(\{(\{(\bullet, \varphi')\}, \varphi')\}, \varphi), (\{(\{(\bullet, \varphi')\}, \varphi')\}, \varphi'), (\emptyset, \varphi), (\emptyset, \varphi')\}}_{⑤}\}$$

each ordered by inclusion. By contrast, the codomain $C_i'$ of $d_i$ is defined as $C_0' = C_0$, $C_i' = \mathcal{P}(C_i \times \Phi)$ for $i = 1, 2, 3$.

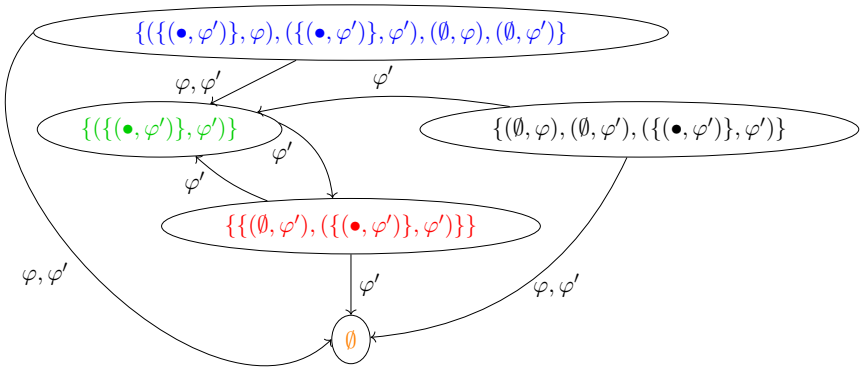
By comparing the columns for each state we can determine which states are bisimilar. The partitions are divided as follows (where $X_i$ denote the entries at the $i^{\text{th}}$ iteration):

$$X_0 = \{\{x, y, z, x', y', z'\}\} \quad X_1 = \{\{x, x'\}, \{y, y'\}, \{z, z'\}\}$$
$$X_2 = X_3 = \{\{x\}, \{x'\}, \{y, y'\}, \{z, z'\}\}.$$

To obtain the greatest conditional bisimulation from $e_2$ (or $e_3$), we need to compare individual entries of each table. We can identify the greatest bisimulation as $\{R_\varphi, R_{\varphi'}\}$, where (written as equivalence classes)

$$R_\varphi = \{\{z, z'\}, \{x\}, \{x'\}, \{y, y'\}\} \quad R_{\varphi'} = \{\{x, x'\}, \{y, y'\}, \{z, z'\}\}.$$

Additionally, it is possible to derive the minimal coalgebra that was identified using the minimisation algorithm, which is of the form $(E_2, m_3 \circ \iota)$ where $\iota \colon E_2 \to \widehat{\mathcal{V}}(E_2)$ is the arrow witnessing termination of the algorithm. The minimisation has the following form:




Note that, if there was no order on $\Phi$, $x$ and $x'$ would be found equivalent under $\varphi$, because without upgrading, $x$ and $x'$ behave the same for $\varphi$: Both can do exactly one step, reaching either of $y, z$ or $z'$, respectively, but in none of these states any additional steps are possible in the condition $\varphi$.

One can observe that both $x$ and $x'$ get mapped under $\varphi'$ to the red state (second from bottom of the diagram), but under $\varphi$, the state $x$ gets mapped to the blue state (top state in the diagram), whereas $x'$ gets mapped to the black state (right-most state in the diagram).

**Remark 6.17.** In [BKKS17] we have also given a matrix multiplication algorithm for minimising CTSs. This algorithm is similar to applying Algorithm 6.11, but working conceptually in $\mathsf{Kl}(T)$ rather than $\mathsf{Kl}(\_^\Phi)$.

Since we have seen that both categories are isomorphic, the coalgebraic representation of a LaTS can be determined by applying the given isomorphism to the representation of a CTS. We obtain the following arrow for any state $x \in X$, action $a \in A$ and set of pairs $Y \subseteq (X \times \Phi)^A$:

$$f(x)(Y)(a) = \{\psi \in \Phi \mid \forall \varphi' \leq \psi, x' \in X : x \xrightarrow{a,\varphi'} x' \Rightarrow (x', \varphi') \subseteq Y(a)\}.$$

Similarly, we can also characterise (pseudo-)factorisation in $\mathsf{Kl}(T)$. We could factorise an arrow by converting it to a $\mathsf{Kl}(\_^\Phi)$-arrow and factorising that arrow, then translating

it back to $\mathsf{Kl}(T)$. Since we have already seen that factorising in $\mathsf{Kl}(\_^{\Phi})$ basically means to exclude all states from the codomain of the arrow that are not in the image of any pair of states and alphabet symbol, this boils down to finding out when a state in a $\mathsf{Kl}(T)$-arrow will be identified as redundant in $\mathsf{Kl}(\_^{\Phi})$. So let $f\colon X \nrightarrow Y$ be a $\mathsf{Kl}(T)$-arrow, then $f(x) = b \in (Y \to \mathbb{L})^*$. An element $y \in Y$ will occur in the image of $f(x)$ if there is an irreducible element $\varphi \in \mathcal{J}(\mathbb{L}) = \Phi$ such that $y$ is the smallest element of $Y$ with $b(y) \geq \varphi$. This is the case if $\bigsqcup\{b(y') \mid y' < y\} \neq b(y)$. So, by factorising an arrow in $\mathsf{Kl}(T)$ we eliminate all states $y$ such that $\bigsqcup\{f(x)(y') \mid y' < y\} = b(y)$ for all $x \in X$. Hence, $f = e \circ m$ where $e\colon X \nrightarrow Y'$ and $Y' \subseteq Y$ is the subset of $Y$ that remains after the elimination.

This enables us to execute the algorithm. The relation to the matrix multiplication method is discussed in more detail in [Küp17]. In particular it can be shown that both variants terminate after the same number of iterations.

## 7. Conclusion, Related and Future Work

In retrospect, the Kleisli categories for the lattice monad and the reader monad are equivalent, providing an analogue to the Birkhoff duality between lattices and partially ordered sets. This duality also reflects the duality between a CTS and a LaTS. We investigated two different functors which can be used to model CTSs without upgrades and general CTSs, respectively, in such a way that behavioural equivalence is conditional bisimulation. Though CTSs without upgrades can be modelled using just $\mathcal{P}(\_)^A$, this functor can not be employed for non-discrete orders, i.e., in the case where upgrades are present. When considering upgrades, the individual versions cannot be considered purely a side effect and must instead be observed, which leads to the requirement of making the versions explicit in a way and to our choice of the functor $\mathcal{P}(\_ \times \Phi)^A$.

The Kleisli category for the reader monad has a pseudo-factorisation structure that makes it possible to use a result from [ABH+12] to compute the greatest conditional bisimulation using a final chain-based algorithm for both functors.

Our work obviously stands in the tradition of the work in [ABH+12] and [KK14]. In a broader sense, the modelling technique of using Kleisli categories to obtain the "right" notion of behavioural equivalence goes back to previous work in [HJS07, PT99], where non-deterministic branching of NFA was masked by the use of a Kleisli category (over $\mathsf{Set}$ in this case) to obtain language equivalence as behavioural equivalence rather than bisimulation.

Modelling new types of systems and their behaviour coalgebraically is an ongoing field of research, as evident by recent work for instance by Bonchi et al. on decorated traces [BBC+16], Hermanns et al. on probabilistic bisimulation [HKK14] or Latella et al. on labelled state-to-function transition systems [LMdV15].

System models that can handle various software products derived from a common base are of particular interest in the field of software product lines. Featured transition systems (FTSs) are conceptually the closest to CTSs and can in fact be simulated by CTSs in a rather straightforward way. A featured transition system is defined as a labelled transition system where each transition is guarded by a feature from a common set of features. A given FTS evolves at follows: first, a set of features (which corresponds to a condition in a CTS) is chosen and transitions are activated or deactivated accordingly, then, the FTS evolves just like a labelled transition system. By choosing for the set of conditions the powerset of all features, ordered discretely, one can simulate FTSs via CTSs (cf. [BKKS17]). Due

to the upgrading aspect of CTSs, the same does not hold the other way around. Similar systems to CTSs have been studied for instance by Cordy et al. [CCP+12] and Kupferman [KL10]. FTSs in particular have been an active field of study in the past years, with various similar, yet not identical definitions being conceived in various lines of work. Classen et al. [CHS+10], as well as Atlee et al. [AFL15] and Cordy et al. [CCH+13] have worked, among many others, on FTSs and the accompanying feature diagrams.

In the future, we want to characterise conditional bisimulation via operational semantics and an appropiate logic. Furthermore, we are interested in analysing different properties of CTSs rather than bisimulation, in particular we are interested in a notion of weak bisimilarity. For this purpose, we will consider adapting a path-based approach similar to the one present in [BK17] to the Kleisli category of the reader monad.

In this paper, we have already taken steps to adapt the notion of conditional bisimilarity to a coalgebraic setting, making it independent of the concrete model and functor under investigation. We plan to investigate whether the notion of conditions (or software products) can be introduced for various state-based system models, for instance for probabilistic systems. That is, we are interested in combining the (sub)distribution functor with our monads, in order to coalgebraically model and analyse families of probabilistic systems in a unified way. From the point of view of software product lines, this could be an entry point to a quantitative analysis of software product lines, rather than a purely qualitative one.

**Acknowledgements.** The authors thank Stefan Milius for fruitful discussions.

## References

[ABH+12] J. Adámek, F. Bonchi, M. Hülsbusch, B. König, S. Milius, and A. Silva. A coalgebraic perspective on minimization and determinization. In *Proc. of FOSSACS '12*, pages 58–73. Springer, 2012. LNCS/ARCoSS 7213.

[AFL15] J. M. Atlee, U. Fahrenberg, and A. Legay. Measuring behaviour interactions between product-line features. In *Proc. of Formalise '15*, pages 20–25, Piscataway, NJ, USA, 2015. IEEE Press.

[AHS90] J. Adámek, H. Herrlich, and G.E. Strecker. *Abstract and Concrete Categories – The Joy of Cats*. Wiley, 1990.

[AK95] Jiří Adámek and Václav Koubek. On the greatest fixed point of a set functor. *Theoretical Computer Science*, 150:57–75, 1995.

[BBC+16] F. Bonchi, M. Bonsangue, G. Caltais, J.J.M.M. Rutten, and A. Silva. A coalgebraic view on decorated traces. *Mathematical Structures in Computer Science*, 26(7):1234–1268, 2016.

[Bir37] G. Birkhoff. Rings of sets. *Duke Mathematical Journal*, 3(3):443–454, 1937.

[BK17] H. Beohar and S. Küpper. On path-based coalgebras and weak notions of bisimulation. In *Proc. of CALCO '17*, 2017. to appear.

[BKKS17] H. Beohar, B. König, S. Küpper, and A. Silva. Conditional transition systems with upgrades. *11th International Symposium on Theoretical Aspects of Software Engineering (TASE 2017)*, 2017. Full version available at `https://arxiv.org/abs/1706.02526`.

[CCH+13] M. Cordy, A. Classen, P. Heymans, A. Legay, and P.-Y. Schobbens. Model checking adaptive software with featured transition systems. In *Assurances for Self-Adaptive Systems*, volume 7740 of *LNCS*, pages 1–29. Springer, 2013.

[CCP+12] M. Cordy, A. Classen, G. Perrouin, P.-Y. Schobbens, P. Heymans, and A. Legay. Simulation-based abstractions for software product-line model checking. In *34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland*, pages 672–682, 2012.

[CCS+13] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, and J.-F. Raskin. Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking. *IEEE Trans. Softw. Eng.*, 39(8):1069–1089, August 2013.

[CHS+10] A. Classen, P. Heymans, P.-Y. Schobbens, A. Legay, and J.-F. Raskin. Model checking lots of systems: Efficient verification of temporal properties in software product lines. In *Proc. of ICSE'10*, pages 335–344, NY, USA, 2010. ACM.

[CN01] P. Clements and L. M. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[DP02] B. A. Davey and H. A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 2002.

[HJS07] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, Volume 3, Issue 4, November 2007.

[HKK14] H. Hermanns, J. Krcál, and J. Kretínský. Probabilistic bisimulation: Naturally on distributions. *CoRR*, abs/1404.5084, 2014.

[KK14] B. König and S. Küpper. Generic partition refinement algorithms for coalgebras and an instantiation to weighted automata. In *Proc. of TCS '14*, IFIP AICT, pages 311–325. Springer, 2014. LNCS 8705.

[KL10] O. Kupferman and Y. Lustig. Latticed simulation relations and games. *International Journal of Foundations of Computer Science*, 21(02):167–189, 2010.

[Küp17] S. Küpper. *Behavioural Analysis of Systems with Weights and Conditions – Coalgebraic and Algorithmic Perspectives on Behavioural Analysis*. PhD thesis, Universität Duisburg-Essen, 2017.

[LMdV15] D. Latella, M. Massink, and E.P. de Vink. Bisimulation of labelled state-to-function transition systems coalgebraically. *Logical Methods in Computer Science*, 11(4), 2015.

[MLM92] S. Mac Lane and I. Moerdijk. *Sheaves in geometry and logic: a first introduction to topos theory*. Universitext. Springer, 1992.

[Mog89] E. Moggi. An abstract view of programming languages. *Edinburgh Univ., Dept. of Comp. Sci (1989) Lecture Notes for course CS 359, Stanford Univ.*, 1989.

[Mog91] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55 – 92, 1991.

[Mul94] P.S. Mulry. Lifting theorems for Kleisli categories. In Stephen Brookes, Michael Main, Austin Melton, Michael Mislove, and David Schmidt, editors, *Mathematical Foundations of Programming Semantics: 9th International Conference New Orleans, LA, USA, April 7–10, 1993 Proceedings*, pages 304–319, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[Par81] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.

[PG14] M. Piróg and J. Gibbons. The coinductive resumption monad. *Electronic Notes in Theoretical Computer Science*, 308:273 – 288, 2014. Proceedings of the 30th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXX).

[PT99] J. Power and D. Turi. A coalgebraic foundation for linear time semantics. In M. Hofmann, D. Pavlović, and G. Rosolini, editors, *Proc. 8th CTCS Conf.*, volume 29 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1999.

[Rut00] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3 – 80, 2000. Modern Algebra.