A Uniform Framework for Timed Automata

Tomasz Brengos[1] **and Marco Peressotti**[2]

1 **Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland**
t.brengos@mini.pw.edu.pl

2 **Department of Mathematics, Computer Science, and Physics, University of Udine, Italy**
marco.peressotti@uniud.it

Abstract

Timed automata, and machines alike, currently lack a general mathematical characterisation. In this paper we provide a uniform coalgebraic understanding of these devices. This framework encompasses known behavioural equivalences for timed automata and paves the way for the extension of these notions to new timed behaviours and for the instantiation of established results from the coalgebraic theory as well. Key to this work is the use of *lax functors* for they allow us to model time flow as a context property and hence offer a general and expressive setting where to study timed systems: the index category encodes "how step sequences form executions" (e.g. whether steps duration get accumulated or kept distinct) whereas the base category encodes "step nature and composition" (e.g. non-determinism and labels). Finally, we develop the notion of *general saturation* for lax functors and show how equivalences of interest for timed behaviours are instances of this notion. This characterisation allows us to reason about the expressiveness of said notions within a uniform framework and organise them in a spectrum independent from the behavioural aspects encoded in the base category.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Coalgebras, lax functors, general saturation, timed behavioural equivalence, timed language equivalence

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2016.26

1 Introduction

Timed automata, and machines alike, are abstract devices used in the modelling and verification of real-time dynamical systems and recipients of much attention, both in terms of theoretical and practical developments [2, 17, 9, 19]. Despite decades long efforts, a general and mathematical characterisation of all these devices is missing. A uniform account of these formalisms would provide guidelines for developing new kinds of timed systems starting from existing notions of computations and, from a more foundational point of view, would allow a cross-fertilizing exchange of definitions, notions, and techniques.

The theory of coalgebras has been recognized as a good context for the study of concurrent and reactive systems [22]: systems are represented as maps of the form $X \to BX$ for some suitable *behavioural functor* B. By changing the underlying category and functor a wide range of cases are covered, from traditional LTSs to systems with I/O, quantitative aspects, probabilistic distribution, and even systems with continuous state. Frameworks of this kind provide great returns from a theoretical and a practical point of view, since they prepare the ground for general results and tools which can be readily instantiated to various cases, and moreover they help us to discover connections and similarities between apparently different


© Tomasz Brengos and Marco Peressotti;
licensed under Creative Commons License CC-BY

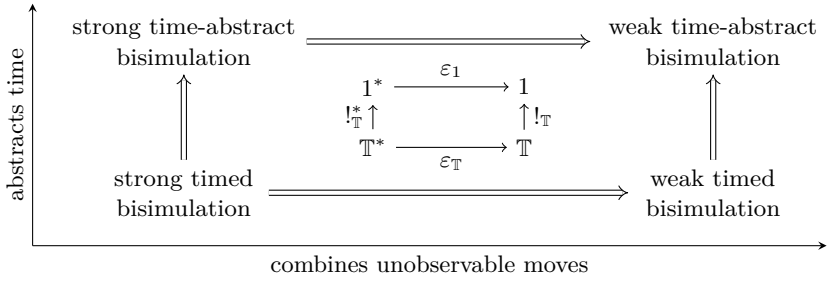


Figure 1 Monoid morphisms and the corresponding spectrum of behavioural equivalences.

notions. Among the several valuable results offered by the coalgebraic approach we mention general accounts of bisimulation [1, 26], structural operational semantics [27, 21], trace equivalence [12], minimization [3], determinisation [23] and up-to techniques [4].

Recent works [8, 6, 7] extended the theory of coalgebras with a general understanding of coalgebras with unobservable moves and their weak bisimulations. As shown in loc. cit. systems with internal steps should be modelled as coalgebras whose type is a monad. This allows us to view coalgebras as endomorphisms in suitable Kleisli categories, hence, allowing mutual composition. At the heart of weak bisimulation of such systems lies the notion of *saturation*. Intuitively, it can be understood as a reflexive and transitive closure of a system. Already, at this coalgebraic level, the so-called *lax functors* start to emerge. Indeed, the reflexive and transitive morphisms can be seen as lax functors whose domain category is the terminal category [7]. Additionally, the results of [6] demonstrate that the coalgebraic saturation arises as a consequence of an adjoint situation between certain two categories of lax functors. All these observations suggest that lax functors and adjunctions between lax functor categories lie at the heart of different behavioural equivalences that take into account accumulation and abstraction of certain portions of data.

In this paper we embrace and extend the lax functorial setting as a general environment where to model systems dynamics. In the approach we propose:

- the index category models those aspects of the computations under scrutiny associated with the computation *context*, like time flow;
- the base category models *effects* like non-determinism or unobservable moves.

Because systems considered in this work have a flat[1] state space we can safely restrict to index categories with one object i.e. monoids.

We extend the theory of saturation by allowing for monoid homomorphisms whose codomain is not the final one-object monoid thus introducing the so called *general saturation* (for lax functors on a monoid category). This theory extends the uniform definition of strong and weak behavioural equivalences given by (simple) saturation offering a greater control of which computational aspects are abstracted away. The resulting notion of behavioural equivalence is parameterised by monoid congruences and naturally forms a *spectrum* reflecting their *discriminating power*. This spectrum reflects the inclusion ordering on congruences with coarser congruences yielding coarser notions of behavioural equivalence. This remarkable correspondence allows for reasoning about the expressiveness of these definitions by means of simple diagrams of monoid homomorphisms (cf. Figure 1). Besides from finer to coarser, this spectrum can be organised along two orthogonal dimensions which intuitively correspond to:

[1] We say that a class of systems has flat state spaces when there are no archetypal rôles associated to states that prescribe restrictions on their possible behaviours, as opposed i.e., to alternating games.

- collecting effects modelled in the base category (e.g. τ-actions) and
- abstracting from aspects modelled in the index category (e.g. time).

The former is determined by the counit $\varepsilon\colon (-)^* \to \mathcal{I}d$ of the free-monoid adjunction whereas the latter by arbitrary monoid morphisms and their extension to free monoids as exemplified by the inner diagram in Figure 1.

Synopsis and related work Our paper is closely related to the research presented in [5, 7, 8, 6] with the emphasis laid on [6]. Indeed, Brengos' [6], which is highly motivated by Sobociński's work on relational presheaves (i.e. lax functors whose codomain category is the category of sets and relations) and their saturation [25], presents the lax functorial framework as a natural extension of coalgebraic weak bisimulation and saturation studied in [7, 8]. The main focus of [6] is on lax functorial weak bisimulation and reflexive and transitive saturation. It is worth mentioning that in loc. cit. the author already pointed out that timed transition systems and their weak behavioural equivalence can be modelled in the lax functorial setting. Our paper extends these results in a systematic way by:

- describing the categorical framework which pinpoints the relation between timed automata and their semantics (Section 2.2),
- presenting the concept of general saturation and the family of behavioural equivalences associated with it (Section 2.3),
- capturing a much wider spectrum of language and behavioural equivalences (Section 3).

2 A coalgebraic account of timed automata and their semantics

In this section we introduce a general framework for timed automata and systems alike. The key ingredient is to separate time flow from other aspects of the computation (like non-determinism) and model the dependency of the latter from the former by means of (lax) functors. We remark that, although we use timed automata as a reference example, this framework can accommodate any notion of resources and effects as long as they can be modelled by some monoid M and by a (suitably enriched) category K, respectively.

We assume the reader to be familiar with basic 1-category and 2-category theory notions and refer to [20, 18] for a thorough introduction.

2.1 Unobservable moves, acceptance and non-determinism

The behaviour associated to the classical notion of timed automata arise from non-determinism, unobservable moves, and acceptance with the latter being relevant when language equivalence is considered. All these computational effects are described briefly below and are captured by Kleisli categories of well-known monads we will use as running examples.

The first example we discuss is the powerset monad $\mathcal{P}$ whose Kleisli category $\mathcal{K}l(\mathcal{P})$ consists of sets as objects and maps of the form $X \to \mathcal{P}Y$ as morphisms between X and Y and, for each $f\colon X \to \mathcal{P}Y$ and $g\colon Y \to \mathcal{P}Z$, the composite $g \circ f\colon X \to \mathcal{P}Z$ is:

$$g \circ f(x) = \bigcup g(f(x)) = \{z \mid z \in g(y) \text{ and } y \in f(x)\}.$$

Let $\Sigma_\tau \triangleq \Sigma + \{\tau\}$. The Set-endofunctor $\mathcal{P}^\Sigma \triangleq \mathcal{P}(\Sigma_\tau \times \mathcal{I}d)$ whose coalgebras model labelled transition systems with unobservable moves [22] is a monad [7], herein the *LTS monad*. The Kleisli category $\mathcal{K}l(\mathcal{P}^\Sigma)$ has sets as objects and maps $X \to \mathcal{P}(\Sigma_\tau \times Y)$ as morphisms from X to Y. For $f\colon X \to \mathcal{P}^\Sigma(Y)$ and $g\colon Y \to \mathcal{P}^\Sigma(Z)$ the composite $g \circ f \in \mathcal{K}l(\mathcal{P}^\Sigma)$ is

$$g \circ f(x) = \{(\sigma, z) \mid x \xrightarrow{\sigma}_f y \xrightarrow{\tau}_g z \text{ or } x \xrightarrow{\tau}_f y \xrightarrow{\sigma}_g z\},$$

where $x \xrightarrow{\sigma}_f y$ denotes $(\sigma, y) \in f(x)$. The above Kleisli category (see [7, 8] for a discussion) lets us consider non-deterministic systems with unobservable moves (a.k.a. τ-actions) as endomorphisms and allows their mutual composition. We refer the interested reader to [5, 7] for further details on how to extend other computational effects besides non-determinism with unobservable moves.

Finally, the endofunctor $\mathcal{P}^{\Sigma}_{\checkmark} \triangleq \mathcal{P}(\Sigma_\tau \times \mathcal{I}d + \{\checkmark\})$ whose coalgebras model the behaviour of non-deterministic automata with ε-moves (or ε-NA in short) [11, 24] also carries a monadic structure [7, Example 4.5] (here and in loc. cit., the label ε is represented by the label τ). The composition in its Kleisli category is given as follows. For two morphisms $f\colon X \to \mathcal{P}^{\Sigma}_{\checkmark}(Y)$ and $g\colon Y \to \mathcal{P}^{\Sigma}_{\checkmark}(Z)$ the value of their composition $g \circ f$ in $\mathcal{K}l(\mathcal{P}^{\Sigma}_{\checkmark})$ on $x \in X$ is:

$$\{(\sigma, z) \mid x \xrightarrow{\sigma}_f y \xrightarrow{\tau}_g z \text{ or } x \xrightarrow{\tau}_f y \xrightarrow{\sigma}_g z\} \cup \{\checkmark \mid \checkmark \in f(x) \text{ or } x \xrightarrow{\tau}_f y \text{ and } \checkmark \in g(y)\}.$$

Morphisms of this category model non-deterministic computations with unobservables and accepting states (for a morphism $f\colon X \to \mathcal{P}^{\Sigma}_{\checkmark}Y$ any such state $x \in X$ is specified by $\checkmark \in f(x)$, which we will often denote as $x\downarrow$). It is important to note that $\mathcal{P}^{\Sigma}$ is a submonad of $\mathcal{P}^{\Sigma}_{\checkmark}$. However, since the terminating states in classical timed automata and their semantics are an extra feature [2] we decide to consider these two examples of monads separately. Indeed, the Kleisli category for the LTS monad will be our main example throughout Section 2.2, 2.3 and 3.1. Our focus will move onto $\mathcal{K}l(\mathcal{P}^{\Sigma}_{\checkmark})$ in Section 3.2, where we discuss finite trace equivalence and sets of accepted words.

The essence of non-determinism of these three examples lies in the fact that their Kleisli categories are suitably order enriched in a natural point-wise manner:

$$f \leq g \overset{\triangle}{\iff} f(x) \subseteq g(x) \quad \forall x \in X$$

with their hom-posets admitting arbitrary joins. Composition in $\mathcal{K}l(\mathcal{P})$ and $\mathcal{K}l(\mathcal{P}^{\Sigma})$ preserves them, whereas in $\mathcal{K}l(\mathcal{P}^{\Sigma}_{\checkmark})$ it only preserves non-empty ones [7]. Joins of morphisms, which abstractly model non-deterministic choice, will play a key rôle in saturation.

▶ Remark. It is an easy exercise to prove that these three Set-based monads are commutative strong monads [16]. It means that the monoidal structure $(\mathsf{Set}, \times, 1)$ extends to all three Kleisli categories above in a natural manner.

2.2 Timed automata in K and their semantics

Recall from [2] that a timed automaton, a.k.a. *time transition table*, over an alphabet Σ and with access to C-many clocks is essentially a set of locations L and a relation on source and target locations, symbols, and expressions for describing clock constraints and resets. However, the actual semantics of *clock expressions* is not part of the definition of timed transition tables but rather of their semantics as *timed transition systems* (herein TTS) which is covered below. Hence, a timed automaton is essentially a morphism $L \to \mathcal{E} \otimes L$ in $\mathsf{K} = \mathcal{K}l(T)$ (where T is $\mathcal{P}^{\Sigma}_{\checkmark}$ or $\mathcal{P}^{\Sigma}$, depending whether acceptance is necessary, and $\otimes$ is $\times$) or, more generally, a $(\mathcal{E} \otimes \mathcal{I}d)$-coalgebra for some given object of (clock) *expressions* $\mathcal{E} \in \mathsf{K}$ for a monoidal category $(\mathsf{K}, \otimes, I)$.

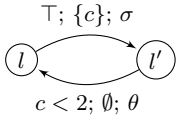
The interaction between timed automata and time flow is mediated by (a fixed and finite set of) clocks which an automaton can interact with by means of the clock expressions associated to its transitions. Clocks can be abstracted as an object $\mathcal{V} \in \mathsf{K}$ representing *clock configurations* (a.k.a. clock valuations) together with:

- a morphism $\mathsf{eval}\colon \mathcal{V} \otimes \mathcal{E} \to \mathcal{V}$ specifying the effect of expressions and

- a $\mathbb{T}$-indexed family $\{\mathsf{flow}_t \colon \mathcal{V} \to \mathcal{V}\}_{t\in\mathbb{T}}$ describing the effect of time flow and such that $\mathsf{flow}_0 = id_{\mathcal{V}}$ and, for all $t, t' \in \mathbb{T}$, $\mathsf{flow}_{t+t'} = \mathsf{flow}_t \circ \mathsf{flow}_{t'}$ with $\mathbb{T} = (\mathbb{T}, +, 0)$ being the monoid of time, e.g. $\mathbb{T} = ([0, \infty), +, 0)$.

In the case of timed automata (with clocks in C) the object of configurations is the set $\mathcal{V}$ of all functions $v \colon C \to \mathbb{T}$ and the object of expressions is the set $\mathcal{E}$ of pairs (γ, δ) where γ is a subset of C and δ is a syntactic expression generated by the grammar $\delta ::= c \leq r \mid r \leq c \mid \neg\delta \mid \delta \wedge \delta$ where c is a clock and the r is a non-negative rational number (cf. [2, Def. 3.6]).

▶ **Example 2.1.** Consider the timed transition table from [2, Ex. 3.4] depicted below: where $\top$ can be seen as a short hand for $0 \leq c$ which is always satisfied. Let $C = \{c\}$ and $\Sigma = \{\sigma, \theta\}$. Intuitively, the edge from l to l' describes a transition that can be performed provided the input character is σ and resets c as its side effect. The other transition assumes $c < 2$ input θ and does not reset c. This automaton is equivalent to the $\mathcal{E} \otimes \mathcal{I}d$-coalgebra α on $\{l, l'\}$ such that:



$$\alpha(l) = \{(\sigma, (\{c\}, \top), l'))\} \qquad \alpha(l') = \{(\theta, ((\emptyset, c < 2), l))\}.$$

Time flow is given, on $t \in \mathbb{T}$, as $\mathsf{flow}_t(v) = \{(\tau, \lambda c : C.v(c) + t)\}$ and expression evaluation as

$$\mathsf{eval}(v, (\gamma, \delta)) = \begin{cases} \{(\tau, v[\gamma := 0])\} & \text{if } v \vDash \delta \\ \emptyset & \text{otherwise} \end{cases} \quad \text{where} \quad v[\gamma := 0](c) = \begin{cases} 0 & \text{if } c \in \gamma \\ v(c) & \text{otherwise} \end{cases}$$

and $v \vDash \delta$ is the obvious interpretation of the boolean expression obtained by replacing every c in δ with $v(c)$.

▶ Remark. At a first glance τs and singletons may sound strident, especially given how the semantics of timed automata is presented in [2]. However, their meaning is precise and plays a crucial rôle in capturing how eval and flow interact with timed automata in defining their semantics as timed transition systems—as will be made clear below. In fact, they arise from the unit of $\mathcal{P}^{\Sigma}$ meaning that when composed the only computational effect (modelled in K and) carried by eval and flow is to prevent the firing of any transition not enabled by the current clock configuration.

Given the semantics for expressions and (time) flow, every timed automaton yields a *timed transition system* (TTS) that is a transition system with labels[2] in $\Sigma_\tau \times \mathbb{T}$ or, equivalently, a $\mathbb{T}$-indexed family of LTSs i.e. endomorphisms in $\mathsf{K} = \mathcal{K}l(\mathcal{P}^{\Sigma})$:

$$\frac{X \to \mathcal{P}(\Sigma_\tau \times \mathbb{T} \times X)}{\mathbb{T} \to \mathcal{P}(\Sigma_\tau \times X)^X}$$

Albeit the two presentations are equivalent for TTSs, we adopt the latter since:

- As noted in [17], the correspondence does not hold for arbitrary timed behaviours e.g. the convex-set semantics of Segala systems assume probability distribution supports to be (finitely) bounded (cf. [5, 13]) whereas a single entry of a timed transition table can easily yield uncountably many transitions in the associated TTS.
- Time flow is an aspect of the computational context instead of an observation and the latter representation allows us to separate the rôle of time from non-deterministic computations modelled in the base category K—along the lines of the lax functor approach.

[2] The definition of TTS may vary: some authors (e.g. [2]) consider transitions to be labelled by pairs (σ, t) of consumed symbols and time (durations) whereas others (e.g. [19]) consider "duration-less" *discrete transitions* (i.e. labelled by $\sigma \in \Sigma$) and "symbol-less" *time transitions* (i.e. labelled by $t \in (0, \infty)$). By introducing a distinguished symbol τ the two approaches are uniformly covered by a single model where labels are pairs in $\Sigma_\tau \times \mathbb{T}$; duration-less and symbol-less transitions become $(\sigma, 0)$ and (τ, t), respectively.

In general, given eval and flow, every $\mathcal{E}\otimes\mathcal{I}d$-coalgebra, i.e. every morphism $\alpha\colon L \to \mathcal{E}\otimes L$ in K induces its *semantics*, i.e. a $\mathbb{T}$-indexed family $\{\alpha_t\colon \mathcal{V}\otimes L \to \mathcal{V}\otimes L\}_{t\in\mathbb{T}}$ where each endomorphism $\alpha_t \in \mathsf{K}$ is defined as:

$$\mathcal{V}\otimes L \xrightarrow[\mathsf{flow}_t\otimes\alpha]{} \mathcal{V}\otimes\mathcal{E}\otimes L \xrightarrow[\mathsf{eval}\otimes id_L]{} \mathcal{V}\otimes L \qquad \mathcal{V}\otimes L \xrightarrow{\ \alpha_t\ } \mathcal{V}\otimes L$$

From the definition of α_t it is clear how any computational effect described by eval and flow interact with α. In particular, when $\mathsf{K} = \mathcal{K}l(\mathcal{P}^{\Sigma})$ and α models a timed automaton, the above definition readily expands as follows:

$$\alpha_t(v,l) = \{(\sigma,v',l') \mid (\sigma,(\gamma,\delta),l') \in \alpha(l),\ v_t = \lambda c.v(c)+t,\ v_t \vDash \delta, \text{ and } v' = v_t[\gamma := 0]\} \quad (1)$$

highlighting the contribution of effects in the definition of expression and flow semantics for timed automata. Clearly, (1) precisely defines the timed transition system associated to the given automaton α. This correspondence can be easily checked when the presentation from [2] is used: consider the LTS with labels in $\Sigma_\tau \times \mathbb{T}$ given by

$$(v,l) \xrightarrow{(\sigma,t)} (v',l') \iff^{\triangle} (\sigma,v',l') \in \alpha_t(v,l)$$

then recall from [2, Def. 3.8] that

$$(l,v) \xrightarrow{(\sigma,t)} (l',v') \iff^{\triangle} \exists (l,\delta,\gamma,\sigma,l') \in T_\alpha \text{ s.t. } v+t \vDash \delta \text{ and } v' = (v+t)[\gamma := 0]$$

where $T_\alpha \subseteq (L,\mathcal{E},\Sigma,L)$ is the automaton transition table (cf. [2, Def. 3.7]) and

$$(l,\delta,\gamma,\sigma,l') \in T_\alpha \iff (\sigma,(\gamma,\delta),l') \in \alpha(l).$$

▶ **Example 2.2.** Let α be the coalgebra from Example 2.1, then:

$$\alpha_t(v,l) = \{(\sigma,(v+t)[c := 0],l')\} \qquad \alpha_t(v,l') = \{(\theta,(v+t),l) \mid (v+t) < 2\}.$$

Although $\mathsf{flow}_{t+t'} = \mathsf{flow}_t \circ \mathsf{flow}_{t'}$ we can derive neither $\alpha_{t+t'} \leq \alpha_t \circ \alpha_{t'}$ nor $\alpha_{t+t'} \geq \alpha_t \circ \alpha_{t'}$. This can be routed to clock constraints being as fine grained as being able to single out exact time instants (e.g. $c = 42$) and hence the semantics cannot preserve (not even up-to laxness) the action of time described by flow. Indeed, the notion of TTS does not assume any condition on the time component of transitions (cf. [2, 17]) and the associated notion of execution (or run) maintains transition duration distinct.

There is a 1-1 correspondence between families $\{\alpha_t\}_{t\in\mathbb{T}}$ and (strict) functors $\underline{\alpha}\colon \mathbb{T}^* \to \mathsf{K}$:

$$\underline{\alpha}_{t_1\dots t_n} = \alpha_{t_1} \circ \dots \circ \alpha_{t_n} \text{ for } t_1 \dots t_n \in \mathbb{T}^*$$

where the free monoid $\mathbb{T}^*$ is seen as a single object category: each α_t is $\underline{\alpha}(t)$ for $t\colon * \to *$ and the shared carrier is $\underline{\alpha}(*)$. In order to better illustrate the rôle of the free monoid let $\mathbb{T}$ be the trivial monoid $1 = (\{0\},+,0)$ and recall that $1^* \cong \mathbb{N}$: a functor $\underline{\alpha}\colon 1^* \to \mathsf{K}$ is an endomorphism in K together with all its finite self-compositions (i.e. $\underline{\alpha}(n) = \alpha^n$). As observed in [6] this precisely puts coalgebras with unobservable moves into the (lax) functorial picture as they are endomorphisms in the Kleisli categories for their type monads.

So far we modelled the semantics of timed automata as families of transition systems indexed over time durations i.e. strict functors from $\mathbb{T}^*$ to K. This setting is enough to define "timed" behavioural equivalences by componentwise extension of the "untimed" notion

but, however intuitive it may be, its limited applicability become clear as soon as weak or time-abstract bisimulations are considered. In such cases *lax functors* have to be considered.

In fact, all these notions share a certain pattern: they combine computations abstracting time or unobservable moves (i.e. effects). Since computations of a timed automaton α are described by a functor $\underline{\alpha}\colon \mathbb{T}^* \to \mathsf{K}$, these transformations can be intuitively understood as "turning a functor $\mathbb{T}^* \to \mathsf{K}$ into some functor-like assignment with a different base category". In the remaining of the section we develop a general theory of such transformations we will refer to as *saturations*.

2.3 General saturation for lax functors

In this section we extend the theory of saturation developed in [7, 8, 6] for modelling weak behavioural equivalences for systems with unobservable moves to cope with time in the abstract sense described above. Intuitively, saturation can be understood as a closure of a given system w.r.t. a certain pattern (e.g. reflexive and transitive closure). Akin to loc. cit., the theory is developed in an order enriched setting where the order stems from suitable notion of simulation and non-determinism between systems and provides a notion of approximation between the intermediate steps of the aforementioned closure operation. For timed automata, the ordering is given by pointwise extension of the inclusion order defined by $\mathcal{P}$ (cf. loc. cit.).

Below we assume J is a wide subcategory of an order enriched category K (i.e. a subcategory with all objects from K).

The category of lax functors

Here we focus on recalling the main notions from the theory of order enriched categories and lax functors needed in our paper.

A functor-like assignment π from a category $\mathbb{D}$ to K is called *lax functor* if:

- $id_{\pi D} \leq \pi(id_D)$ for any object $D \in \mathbb{D}$,
- $\pi(d_1) \circ \pi(d_2) \leq \pi(d_1 \circ d_2)$ for any two composable morphisms $d_1, d_2 \in \mathbb{D}$.

Let $\pi, \pi'\colon \mathbb{D} \to \mathsf{K}$ be two lax functors. A family $f = \{f_D\colon \pi D \to \pi' D\}_{D\in\mathbb{D}}$ of morphisms in K is called *lax natural transformation* from π to π' if for any $d\colon D \to D'$ in $\mathbb{D}$ we have $f_{D'} \circ \pi(d) \geq \pi'(d) \circ f_D$. *Oplax functors* and *oplax transformations* are defined by reversing the order in the above. Note that in the more general 2-categorical setting an (op)lax functor and an (op)lax natural transformation are assumed to additionally satisfy extra coherence conditions [18]. In our setting of order enriched categories these conditions are vacuously true, hence we do not list them here.

Let $\mathbb{D}$ be a small category. By $[\mathbb{D}, \mathsf{K}]^J$ we denote the category whose objects are lax functors from $\mathbb{D}$ to K and whose morphisms are oplax transformations with components from J. The category $[\mathbb{D}, \mathsf{K}]^J$ is order enriched with the order on hom-sets given as follows. For $\pi, \pi' \in [\mathbb{D}, \mathsf{K}]^J$ and two oplax transformations $f, f'\colon \pi \to \pi'$ whose components are morphisms in J we define:

$$f \leq f' \iff f_D \leq f'_D \text{ in } \mathsf{K} \text{ for any } D \in \mathbb{D}.$$

Any functor $q\colon \mathbb{D} \to \mathbb{E}$ between small categories induces the *change-of-base functor* $[q, \mathsf{K}]^J\colon [\mathbb{E}, \mathsf{K}]^J \to [\mathbb{D}, \mathsf{K}]^J$ which is given for any object $\pi \in [\mathbb{E}, \mathsf{K}]^J$ and for any oplax transformation $f = \{f_E\colon \pi(E) \to \pi'(E)\}_{E\in\mathbb{E}}$ between $\pi, \pi' \in [\mathbb{E}, \mathsf{K}]^J$ by $[q, \mathsf{K}]^J(\pi) = \pi \circ q$ and $[q, \mathsf{K}]^J(f)_D = f_{q(D)}$.

To keep the paper more succinct, we only focus on $\mathbb{D}$ being a monoid category (i.e. a one-object category). In this case, a monoid $M = (M, \cdot, 1)$ will be often associated with the one-object category it induces. The only object of the category M will be denoted by $*$ and the composition $\circ$ of morphisms $m_1, m_2 \colon * \to *$ for $m_1, m_2 \in M$ is given by: $m_1 \circ m_2 = m_1 \cdot m_2$. In this case, any lax functor $\pi \in [M, \mathsf{K}]^J$ is a family $\pi = \{\pi(m) = \pi_m \colon X \to X\}_{m \in M}$ of endomorphisms in K with a common carrier $X = \pi(*)$ which additionally satisfies [6]:

$$id_X \leq \pi_1 \text{ and } \pi_m \circ \pi_n \leq \pi_{m \cdot n}.$$

An oplax transformation between two lax functors $\pi = \{\pi_m \colon X \to X\}_{m \in M}$ and $\pi' = \{\pi'_m \colon Y \to Y\}_{m \in M}$ in $[M, \mathsf{K}]^J$ is given in terms of an arrow $f \colon X \to Y$ in J which satisfies:

$$f \circ \pi_m \leq \pi'_m \circ f \text{ for any } m \in M.$$

The curious reader is referred to [6] for several examples of these categories.

General saturation

For a functor $q \colon M \to N$ between monoid categories for $(M, \cdot, 1)$ and $(N, \cdot, 1)$ (i.e. a monoid homomorphism) and an order enriched category K we say that K admits *q-saturation* provided that $[q, \mathsf{K}]^J \colon [N, \mathsf{K}]^J \to [M, \mathsf{K}]^J$ admits a left strict 2-adjoint. In order to prove q-saturation admittance below, we work with stronger types of order enrichment on K. We will introduce them now. We say that K is $\mathsf{DCpo}^{\vee}$*-enriched* if it is DCpo-enriched (i.e. each hom-set is a complete order with directed suprema being preserved by the composition) and, additionally, each hom-set admits binary joins (which are *not* assumed to be preserved by the composition). A $\mathsf{DCpo}^{\vee}$-enriched category K is *J-left distributive* if $f \circ (g \vee h) = f \circ g \vee f \circ h$ for any maps $f \in J$ and $g, h \in \mathsf{K}$ with suitable domains and codomains. We define *J-right distributivity* dually. Finally, we say that K is *J-distributive* if it is both, J-left and J-right distributive.

▶ **Theorem 2.3.** *Let $q \colon M \to N$ be a surjective homomorphism and K be J-left distributive (hence $\mathsf{DCpo}^{\vee}$-enriched). Then K admits q-saturation with the left strict 2-adjoint $\Sigma_q : [M, \mathsf{K}]^J \to [N, \mathsf{K}]^J$ given as the identity on morphisms and on any lax functor $\pi \in [M, \mathsf{K}]^J$ as*

$$\Sigma_q(\pi)(*) = \pi(*) \quad \text{and} \quad \Sigma_q(\pi)_w = \bigvee \Pi_w$$

where $w \in N$, $\Pi_w = \bigcup_{n \in \mathbb{N}} \Pi_{w,n}$, $\Pi_{w,0} = \{\pi_{m_1} \vee \ldots \vee \pi_{m_k} \mid q(m_i) = w\}$, and $\Pi_{w,n} = \{t_1 \vee t_2 \vee \ldots \vee t_k \mid t_i \in \Pi_{w_1, n-1} \circ \ldots \circ \Pi_{w_l, n-1}$ and $w_1 \ldots w_l = w\}$.

Proof. Firstly we show that $\Sigma_q(\pi)$ is a well defined lax functor in $[N, \mathsf{K}]^J$. Note that the family $\{\Pi_{w,n}\}_{n \in \mathbb{N}}$ is an ascending family of sets and Π_w is, by surjectivity of q, a non-empty directed set. Moreover, $\Pi_w \circ \Pi_{w'} \subseteq \Pi_{w \cdot w'}$ for any $w, w' \in N$. We have $id \leq \bigvee \Pi_1 \leq \Sigma_q(\pi)_1$ and

$$\Sigma_q(\pi)_w \circ \Sigma_q(\pi)_{w'} = \bigvee \Pi_w \circ \bigvee \Pi_{w'} = \bigvee \Pi_w \circ \Pi_{w'} \leq \bigvee \Pi_{w \cdot w'} = \Sigma_q(\pi)_{w \cdot w'}.$$

This proves the first statement. Now, we will verify that any oplax transformation in $[M, \mathsf{K}]^J$ is mapped onto an oplax transformation in $[N, \mathsf{K}]^J$. Assume $f \circ \pi_m \leq \pi'_m \circ f$ for any $m \in M$. We have $f \circ \Sigma_q(\pi)_w = f \circ \bigvee \Pi_w = \bigvee f \circ \Pi_w \leq \bigvee \Pi'_w \circ f = \Sigma_q(\pi')_w \circ f$, which follows by J-left distributivity, induction and Scott continuity of the composition.

To complete the proof we have to show that for any lax functor $\pi \in [M, \mathsf{K}]^J$ and $\pi' \in [N, \mathsf{K}]^J$ the poset $[M, \mathsf{K}]^J(\pi, [q, \mathsf{K}](\pi'))$ is isomorphic to $[N, \mathsf{K}]^J(\Sigma_q(\pi), \pi')$. Indeed:

$$
\begin{aligned}
&f \circ \pi_m \leq \pi'_{q(m)} \circ f \text{ for any } m \in M \overset{(\dagger)}{\iff} \\
&f \circ (\pi_{m_1} \vee \ldots \vee \pi_{m_k}) \leq \pi'_{q(m)} \circ f \text{ for any } m, m_1 \ldots m_k \in M \text{ and } q(m_i) = q(m) \iff \\
&\forall m \in M, f \circ \bigvee \Pi_{q(m),0} \leq \pi'_{q(m)} \circ f \overset{(\dagger\dagger)}{\iff} \forall m \in M, \forall n \in \mathbb{N}, f \circ \bigvee \Pi_{q(m),n} \leq \pi'_{q(m)} \circ f \\
&\iff \forall m \in M, f \circ \bigvee \Pi_{q(m)} \leq \pi'_{q(m)} \circ f \iff \forall m \in M, f \circ \Sigma_q(\pi)_{q(m)} \leq \pi'_{q(m)} \circ f.
\end{aligned}
$$

The equivalence (†) follows by J-left distributivity of K and (††) by induction on $n \in \mathbb{N}$. ◀

If K comes equipped with an even stronger type of order enrichment then the formula for Σ_q simplifies considerably. Indeed, we have the following (see also [6, Th. 3.14]).

▶ **Theorem 2.4.** *If the hom-posets of K admit arbitrary non-empty suprema which are preserved by the composition then for any $\pi \in [M, \mathsf{K}]^J$ we have:*

$$\Sigma_q(\pi)_w = \bigvee_{m:q(m)=w} \pi_m.$$

Proof. It follows by the fact that in this case we have $\Pi_w = \Pi_{w,0}$. Hence, $\Sigma_q(\pi)_w = \bigvee \Pi_w = \bigvee \Pi_{w,0} = \bigvee_{m:q(m)=w} \pi_m$. ◀

▶ Remark. The notion of lax functorial saturation has already been considered in [6] whenever $N = 1$ is the one-element monoid and $q \colon M \to 1$ the unique homomorphism sending all elements of M to the neutral element of 1. In that case it was simply called saturation and was shown to be directly linked to the notion of coalgebraic weak bisimulation.

In the remaining of this subsection we take $q_1 \colon M \to N_1$, $q_2 \colon N_1 \to N_2$ and $q \colon M \to N$ to be monoid homomorphisms and K to be J-left distributive (hence $\mathsf{DCpo}^\vee$-enriched).

▶ **Theorem 2.5.** *The functor $\Sigma_{q_2} \circ \Sigma_{q_1}$ is (naturally isomorphic to) $\Sigma_{q_2 \circ q_1}$.*

Proof. This statement follows directly by the fact that Σ_{q_1} and Σ_{q_2} are left adjoints and that $[q_1, \mathsf{K}]^J \circ [q_2, \mathsf{K}]^J = [q_2 \circ q_1, \mathsf{K}]^J$. ◀

▶ **Theorem 2.6.** *The functor $\Sigma_q \circ [q, \mathsf{K}]^J$ is the identity on $[N, \mathsf{K}]^J$.*

Proof. It is enough to show that for any $\pi' \in [N, \mathsf{K}]^J$ we have $\Sigma_q(\pi' \circ q) = \pi'$. Take $\pi = \pi' \circ q$ and note that $\Pi_{w,0} = \{\pi_{m_1} \vee \ldots \vee \pi_{m_k} \mid q(m_i) = w\} = \{\pi'_w \vee \ldots \vee \pi'_w \mid q(m_i) = w\} = \{\pi'_w\}$ for $w \in N$. Moreover, since π' is a lax functor we can easily prove by induction that $t \leq \pi'_w$ holds for any $t \in \Pi_{w,n}$. Hence, $\Sigma_q(\pi' \circ q)_w = \bigvee \Pi_w = \pi'_w$ which proves the assertion. ◀

▶ **Theorem 2.7.** *If K is J-distributive then, for any $\pi, \pi' \in [M, \mathsf{K}]^J$ and $f \colon \pi(*) \to \pi'(*) \in J$:*

$$f \circ \pi_m = \pi'_m \circ f \text{ for any } m \in M \implies f \circ \Sigma_q(\pi)_w = \Sigma_q(\pi')_w \circ f \text{ for any } w \in N.$$

Proof. We have:

$$f \circ (\pi_{m_1} \vee \ldots \vee \pi_{m_k}) \overset{J\text{-LD}}{=} f \circ \pi_{m_1} \vee \ldots \vee f \circ \pi_{m_k} = \pi'_{m_1} \circ f \vee \ldots \vee \pi'_{m_k} \circ f \overset{J\text{-RD}}{=} (\pi'_{m_1} \vee \ldots \vee \pi'_{m_k}) \circ f.$$

Hence, for any $f \circ \Pi_{w,0} = \Pi'_{w,0} \circ f$. By induction it follows that $f \circ \Pi_{w,n} = \Pi'_{w,n} \circ f$ for any n. Hence, $f \circ \Pi_w = \Pi'_w \circ f$ which proves the assertion. ◀

2.4 Saturation-based behavioural equivalences

Notions of coalgebraic strong bisimulation have been well captured in the literature [1, 22, 26]. This paper builds on a variant called *kernel bisimulation* i.e. "a relation which is the kernel of a compatible refinement of a system" [26]. However, since coalgebras with internal moves are endomorphisms in suitable Kleisli categories[7], kernel bisimulation has been reformulated in [8] in the language of *behavioural morphisms* for endomorphisms in an arbitrary category K. This choice allows us to streamline the exposition while working at the abstract level of morphism between endomorphisms and accommodate general saturation along the lines of [8]. Let us quickly recall this notion here. We say that $f\colon X \to Y \in J$ is a *behavioural morphism* on an endomorphism $\alpha\colon X \to X \in \mathsf{K}$ provided that there is $\beta\colon Y \to Y \in \mathsf{K}$ such that $f \circ \alpha = \beta \circ f$. If $\mathsf{K} = \mathcal{K}l(T)$ for a monad T on C and J is the category whose objects are those of C and whose morphisms are $\eta_Y \circ f$, where η is the unit of the monad and $f\colon X \to Y$ a morphism of C, then the behavioural morphisms coincide with homomorphisms of T-coalgebras [8]. This approach is adopted in this paper taking lax functors and their saturation into account.

▶ **Definition 2.8.** Let $\pi \in [M, \mathsf{K}]^J$ and $X = \pi(*)$. A morphism $f\colon X \to Y$ in J is called *q-behavioural morphism* on π provided that there is a lax functor $\pi' \in [N, \mathsf{K}]^J$ such that $\pi'(*) = Y$ for any $n \in N$ the following holds:

$$f \circ \Sigma_q(\pi)_n = \pi'_n \circ f.$$

A *q-bisimulation* on π is a relation $R \rightrightarrows \pi(*)$ (i.e. a jointly monic span) in J which is the kernel pair of a q-behavioural morphism on π.

▶ **Theorem 2.9.** *A morphism $f\colon X \to Y$ in J is a q-behavioural morphism on $\pi \in [M, \mathsf{K}]^J$ if and only if there is $\pi' \in [M, \mathsf{K}]^J$ such that $f \circ \Sigma_q(\pi)_n = \Sigma_q(\pi')_n \circ f$ for any $n \in N$.*

Proof. By Theorem 2.6. ◀

Definition 2.8 is a conservative extension of kernel bisimulation. In fact, $\underline{\alpha} \in [M^*, \mathsf{K}]$ describes a family of endomorphisms in K, i.e., a timed system, and all its finite self-composition and since $\Sigma_{id_{M^*}} \cong \mathcal{I}d$, id_{M^*}-bisimulation coincides with kernel bisimulation.

Among the several choices of monoid homomorphisms we focus on two main families: those abstracting effects (e.g. non-determinism) and those abstracting aspects modelled by the context (e.g. time). The first case corresponds to $\varepsilon_M\colon M^* \to M$ where $\varepsilon\colon (-)^* \to \mathcal{I}d$ is the counit forming the free monoid adjunction. By Theorem 2.4, if K admits and preserves arbitrary non-empty joins the functor $\Sigma_{\varepsilon_M}\colon [M^*, \mathsf{K}]^J \to [M, \mathsf{K}]^J$ is given on a system $\underline{\alpha}$:

$$\Sigma_{\varepsilon_M}(\underline{\alpha})_m = \bigvee_{\vec{m} \in \varepsilon_M^{-1}(m)} \underline{\alpha}_{\vec{m}} = \bigvee_{m_1 \cdot \ldots \cdot m_k = m} \alpha_{m_1} \circ \cdots \circ \alpha_{m_k}.$$

Steps in the saturated system are combinations of all sequences in the original one associated with any decomposition of the stage m. In particular, when K models unobservable moves, ε_M-bisimulation can be seen as the weak counterpart of strong behavioural equivalence for systems modelled in $[M^*, \mathsf{K}]^J$, i.e., id_{M^*}-bisimulation. The second case corresponds to $q^*\colon M^* \to N^*$ for some $q\colon M \to N$. The functor $\Sigma_{q^*}\colon [M^*, \mathsf{K}]^J \to [N^*, \mathsf{K}]^J$ is given on a system $\underline{\alpha}$ as:

$$\Sigma_{q^*}(\underline{\alpha})_{n_1 \ldots n_k} = \bigvee_{n_i = q(m_i)} \underline{\alpha}_{m_1 \ldots m_k} = \bigvee_{n_i = q(m_i)} \alpha_{m_1} \circ \cdots \circ \alpha_{m_k}.$$

Steps in the saturated system are combinations of all steps associated to a pre-image through q; in particular, $\Sigma_{q^*}(\underline{\alpha})_n = \bigvee_{n = q(m)} \alpha_m$. When $!_M\colon M \to 1$ is considered, all the information

in M is erased whereas computational effects in K are preserved. From this perspective $!^*_M$-bisimulation can be seen as a conservative generalisation of time-abstract bisimulations (cf. Section 3.1). These two orthogonal directions can be combined as $\varepsilon_N \circ q^*$ or as $q \circ \varepsilon_M$; by naturality of ε the notions of saturation coincide:

$$\Sigma_{\varepsilon_N \circ q^*}(\underline{\alpha})_n = \Sigma_{q \circ \varepsilon_M}(\underline{\alpha})_n = \bigvee_{\substack{n_1 \cdot \ldots \cdot n_k = n \\ n_i = q(m_i)}} \alpha_{m_1} \circ \cdots \circ \alpha_{m_k} = \bigvee_{\substack{n = q(m) \\ m_1 \cdot \ldots \cdot m_k = m}} \alpha_{m_1} \circ \cdots \circ \alpha_{m_k}.$$

When K models unobservable moves and N is 1, $!_{M^*}$-bisimulation can be thought as the weak counterpart of $!^*_M$-bisimulation.

The notion of q-bisimulation for systems modelled in $[M, \mathsf{K}]^J$ arise from some congruence for M and, *vice versa*, each congruence defines a notion of q-bisimulation. The following theorem states that that coarser congruences define coarser notions of bisimulations. In the remainder of this section we assume that $q_1 \colon M \to N_1$ and $q_2 \colon N_1 \to N_2$ are monoid morphisms and K is a J-distributive $\mathsf{DCpo}^\vee$-enriched category.

▶ **Theorem 2.10.** *A q_1-behavioural morphism on π is a $(q_2 \circ q_1)$-behavioural morphism on π.*

Proof. Follows directly by Theorems 2.5 and 2.7. ◀

▶ **Corollary 2.11.** *Let $q_1 \colon M \to N_1$ and $q_2 \colon N_1 \to N_2$ be monoid homomorphisms. For any $\pi \in [M, \mathsf{K}]^J$, if $R \rightrightarrows \pi(*)$ is a q_1-bisimulation on π then it is a $(q_2 \circ q_1)$-bisimulation on π.*

Thus, q-bisimulations for systems modelled in the context of $[M, \mathsf{K}]^J$ form a spectrum where the finest and coarsest notions are defined by $id_M \colon M \to M$ and $!_M \colon M \to 1$, respectively.

3 Application: timed behavioural and language equivalences

This section instantiates the general framework developed in Section 2.4 to timed automata covering semantic equivalences of interest such as (weak) timed and time-abstract bisimulations and (finite) timed and untimed language equivalences.

3.1 Behavioural equivalences

This subsection recovers known notions of behavioural equivalences for timed automata as instances of q-bisimulation thus deriving the spectrum illustrated in Figure 1. Before stating these results, recall from [2, 19] the following definitions of the main bisimulations of interest.

▶ **Definition 3.1** ([2, 19])**.** For a TTS α and an equivalence relation R on its carrier:

- R is a *(strong) timed bisimulation* for α if $(x, y) \in R$ and $x \xrightarrow{(\sigma,t)} x'$ implies that there is $y' \in X$ such that $y \xrightarrow{(\sigma,t)} y'$ and $(x', y') \in R$;
- R is a *(strong) time-abstract bisimulation* for α if $(x, y) \in R$ and $x \xrightarrow{(\sigma,t)} x'$ implies that there are $y' \in X$ and $t' \in [0, \infty)$ such that $y \xrightarrow{(\sigma,t')} y'$ and $(x', y') \in R$;
- R is a *weak timed bisimulation* for α if $(x, y) \in R$ and $x \xRightarrow{(\sigma,t)} x'$ implies that there is $y' \in X$ such that $y \xRightarrow{(\sigma,t)} y'$ and $(x', y') \in R$;
- R is a *weak time-abstract bisimulation* for α if $(x, y) \in R$ and $x \xRightarrow{(\sigma,t)} x'$ implies that there are $y' \in X$ and $t' \in [0, \infty)$ such that $y \xRightarrow{(\sigma,t')} y'$ and $(x', y') \in R$;

where $\to$ and $\Rightarrow$ denote the transition relation of α and the smallest relation such that:

$$\frac{}{x \overset{(\tau,0)}{\Longrightarrow} x} \qquad \frac{x \xrightarrow{(\tau,t)} y}{x \overset{(\tau,t)}{\Longrightarrow} y} \qquad \frac{x \overset{(\tau,t_0)}{\Longrightarrow} x' \quad x' \xrightarrow{(\sigma,t_1)} y' \quad y' \overset{(\tau,t_2)}{\Longrightarrow} y \quad t = t_0 + t_1 + t_2}{x \overset{(\sigma,t)}{\Longrightarrow} y}$$

▶ **Theorem 3.2.** *For a TTS α, an equivalence relation R on its carrier is a timed/weak timed/time-abstract/weak time-abstract bisimulation for α if, and only if, it is a $id_{\mathbb{T}^*}$-/$\varepsilon_{\mathbb{T}}$-/$!^*_{\mathbb{T}}$-/$!_{\mathbb{T}^*}$-bisimulation for $\underline{\alpha}$, respectively.*

▶ **Corollary 3.3.** *For a TTS α and an equivalence relation R on its carrier, the implications illustrated in Figure 1 hold true.*

Proof. By the (inner) commuting diagram in Figure 1, Corollary 2.11 and Theorem 3.2. ◀

Finally, since neither $!^*_{\mathbb{T}}$ factors $\varepsilon_{\mathbb{T}}$ nor *vice versa* it is not possible to derive that, in general, strong time-abstract bisimulations are weak timed ones nor *vice versa*.

3.2 Finite trace equivalence

This subsection focuses on putting the language equivalence of timed systems into the lax functorial setting. Here, we work with $\mathsf{K} = \mathcal{K}l(\mathcal{P}^{\Sigma}_{\checkmark})$- the Kleisli category for the ε-NA monad together with the monoidal structure $(\times, 1)$. A timed coalgebra $\alpha\colon L \to \mathcal{E} \otimes L$ in K is a set map $\alpha\colon L \to \mathcal{P}(\Sigma_\tau \times \mathcal{E} \times L + \{\checkmark\})$ and it represents an ordinary timed automaton with a specified set of terminal states (here, this set is given by $\{l \in L \mid \checkmark \in \alpha(l)\}$). Classically, its semantics is a transition system $\mathcal{V} \times L \to \mathcal{P}(\Sigma_\tau \times \mathbb{T} \times \mathcal{V} \times L + \{\checkmark\})$ over the state-space $\mathcal{V} \times L$, which is obtained from α analogously to (1) taking into account the terminals [2]. Systems of type $X \to \mathcal{P}^{\Sigma}_{\checkmark}(\mathbb{T} \times X)$ will be referred to as *timed transition systems with accepting states* or $\checkmark$*-TTS* in short. However, from our perspective, and following the guidelines of the setting from Section 2.2, the semantics of α is given in terms of a functor $\underline{\alpha}\colon \mathbb{T}^* \to \mathsf{K}$ with $\underline{\alpha}_{t_1,\ldots,t_n} = \alpha_{t_1} \circ \ldots \circ \alpha_{t_n}$, where $\alpha_t\colon \mathcal{V} \times L \to \mathcal{P}^{\Sigma}_{\checkmark}(\mathcal{V} \times L)$ for (v, l) is defined by:

$$\begin{aligned}\alpha_t(v,l) = &\{(\sigma, v', l') \mid (\sigma, l', (\gamma, \delta)) \in \alpha(l),\ v_t = \lambda c.v(c) + t,\ v_t \vDash \delta, \text{ and } v' = v_t[\gamma := 0]\} \\ &\cup \textbf{ if } \checkmark \in \alpha(l) \textbf{ then } \{\checkmark\} \textbf{ else } \emptyset.\end{aligned}$$

Now, in order to obtain finite (weak) (un)timed trace equivalence, we will play with the second parameter that the q-bisimulation takes implicitly into account, namely the base category J. Here, we consider J to be different from Set. We put J to be the Kleisli category $\mathcal{K}l(\mathcal{P})$ for the powerset monad, as it is a subcategory of $\mathcal{K}l(\mathcal{P}^{\Sigma}_{\checkmark})$ with the inclusion functor $(-)^\sharp\colon \mathcal{K}l(\mathcal{P}) \to \mathcal{K}l(\mathcal{P}^{\Sigma}_{\checkmark})$ given, for any X and $f\colon X \to \mathcal{P}Y$ by $X^\sharp = X$ and $f^\sharp(x) = \{(\tau, y) \mid y \in f(x)\}$. Taking the source category of behavioural morphisms to be different from Set is akin to the classical approach towards modelling coalgebraic finite trace equivalence in terms of bisimulation for base categories given by the Kleisli categories for a monadic part of the type functor [12, 14].

Given a state x of a $\checkmark$-TTS we define the following four sets of languages it accepts:

$$\begin{aligned}&T(x) = \{(\sigma_1, t_1)\ldots(\sigma_n, t_n) \in (\Sigma_\tau \times \mathbb{T})^* \mid x \xrightarrow{(\sigma_1,t_1)} \ldots \xrightarrow{(\sigma_n,t_n)} x_n \text{ and } x_n \downarrow\},\\ &UT(x) = \{\sigma_1 \ldots \sigma_n \in \Sigma^*_\tau \mid \exists t_1, \ldots t_n \text{ s.t.} (\sigma_1, t_1)\ldots(\sigma_n, t_n) \in T(x)\},\\ &WT(x) = \{(\sigma_1, t_1)\ldots(\sigma_n, t_n) \in (\Sigma \times \mathbb{T})^* \mid x \overset{(\sigma_1,t_1)}{\Longrightarrow} \ldots \overset{(\sigma_n,t_n)}{\Longrightarrow} x_n \text{ and } x_n \downarrow\},\\ &WUT(x) = \{\sigma_1 \ldots \sigma_n \in \Sigma^* \mid \exists t_1, \ldots t_n \text{ s.t.} (\sigma_1, t_1)\ldots(\sigma_n, t_n) \in WT(x)\}.\end{aligned}$$

▶ **Theorem 3.4.** *Let* $\alpha\colon X \to \mathcal{P}(\Sigma_\tau \times \mathbb{T} \times X + \{\checkmark\})$ *be a* $\checkmark$*-TTS with its induced functor denoted by* $\underline{\alpha}\colon \mathbb{T}^* \to \mathsf{K}$*. A morphism* $f\colon X \to \mathcal{P}Y$ *in* $\mathcal{K}l(\mathcal{P})$ *is a* $id_{\mathbb{T}^*}$*-/*$\varepsilon_{\mathbb{T}}$*-/*$!^*_{\mathbb{T}}$*-/*$\Lambda_{\mathbb{T}^*}$*-behavioural morphism on* $\underline{\alpha}$ *making* $f(x) = f(x')$ *for* $x, x' \in X$ *if, and only if we have respectively:*

$$T(x) = T(x') \;/\; WT(x) = WT(x') \;/\; UT(x) = UT(x') \;/\; WUT(x) = WUT(x').$$

4 Conclusions

In this paper we presented a framework for modelling timed automata and showed how behavioural equivalences of interest can be recovered in a uniform and general way. Moreover, we were able to organise, by means of simple diagrams of monoid homomorphisms, these equivalences in a spectrum on the base of their discriminating power—as Figure 1 exemplifies.

Lax functors are the cornerstone these results build on for they allow us to separate aspects of the computations under scrutiny into those arising from effects (base category) and from the environment (index category). We extended the theory of saturation [6, 8, 7] developing the notion of *general saturation* for lax functors on a monoid category. Akin to saturation cf. loc. cit., this construction offers a uniform definition of behavioural equivalences and a fine control on the computational aspects they abstract away, being these time or unobservable moves. Thanks to its categorical development, this framework can accommodate any computational aspect modelled by some monoid M together with a $\mathsf{DCpo}^{\vee}$-enriched, J-distributive category K. Besides non-deterministic systems we should mention Segala, weighted, probabilistic, nominal, and continuous systems as possible instances for K and refer the curious reader to [6, 8]. Although the categories formed by these systems do not necessarily satisfy left distributivity, in [6, 8] we presented a method to obtain it by moving to a richer category and performing saturation there. This suggests that results described here should readily generalise q-behavioural morphisms to systems which do not necessarily satisfy this paper assumptions.

Although timed processes present some fundamental differences w.r.t. timed automata, we mention the interesting account of these systems and their strong bisimulation by Kick [15]. In loc. cit., time-dependent computations are modelled by a suitable comonad over Set and then combined with other behavioural aspects by means of comonad products. Because of technical difficulties associated with comonad products, this approach appears less flexible when behavioural equivalences beside strong timed bisimulation are considered.

Perhaps the closest work to ours is [10] where a categorical account of timed weak bisimulation is proposed. We remark that loc. cit. relies on open-map bisimulation and is limited to weak timed bisimulation for timed transition systems only.

The categorical characterization of timed automata paves the way for further interesting lines of research. One is to explore the framework expressiveness providing new instances besides timed. Another is to extend the framework with results from the rich theory of coalgebras such as minimization [3], determinisation [23], and up-to techniques [4]. Finally, we believe this, and recent works like [6, 8] suggest the rich and malleable setting of lax functors as a context where to study complex interactions of several computational aspects. To this end, we plan to further develop the general theory of lax functors and saturation.

References

1 Peter Aczel and Nax Mendler. A final coalgebra theorem. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Proc. CTCS*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer, 1989.

2 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.

3 Filippo Bonchi, Marcello M. Bonsangue, Helle Hvid Hansen, Prakash Panangaden, Jan J. M. M. Rutten, and Alexandra Silva. Algebra-coalgebra duality in brzozowski's minimization algorithm. *ACM Trans. Comput. Log.*, 15(1):3, 2014.

4 Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. Coinduction up-to in a fibrational setting. In *CSL-LICS*, pages 20:1–20:9. ACM, 2014.

5 Tomasz Brengos. On coalgebras with internal moves. In Marcello M. Bonsangue, editor, *Proc. CMCS*, Lecture Notes in Computer Science, pages 75–97. Springer, 2014.

6 Tomasz Brengos. Lax functors and coalgebraic weak bisimulation. *CoRR*, abs/1404.5267, 2015.

7 Tomasz Brengos. Weak bisimulation for coalgebras over order enriched monads. *Logical Methods in Computer Science*, 11(2:14):1–44, 2015.

8 Tomasz Brengos, Marino Miculan, and Marco Peressotti. Behavioural equivalences for coalgebras with unobservable moves. *Journal of Logical and Algebraic Methods in Programming*, 84(6):826–852, 2015.

9 Taolue Chen, Tingting Han, and Joost-Pieter Katoen. Time-abstracting bisimulation for probabilistic timed automata. In *TASE*, pages 177–184. IEEE Computer Society, 2008.

10 Natalya Gribovskaya and Irina Virbitskaite. A categorical view of timed weak bisimulation. In *TAMC*, volume 6108 of *Lecture Notes in Computer Science*, pages 443–454. Springer, 2010.

11 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic forward and backward simulations. In *Proc. JSSST Annual Meeting*, 2006.

12 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.

13 Bart Jacobs. Coalgebraic trace semantics for combined possibilistic and probabilistic systems. In *Proc. CMCS*, Electronic Notes in Theoretical Computer Science, pages 131–152, 2008.

14 Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. In Dirk Pattinson and Lutz Schröder, editors, *Proc. CMCS*, volume 7399 of *Lecture Notes in Computer Science*, pages 109–129. Springer, 2012.

15 Marco Kick. *A Mathematical Model of Timed Processes*. Ph.D. dissertation, University of Edinburgh, 2003.

16 Anders Kock. Strong functors and monoidal monads. *Arc. Math.*, 23(1):113–120, 1972.

17 Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282(1):101–150, 2002.

18 Stephen Lack. *A 2-categories companion*. Springer, 2010.

19 Kim G. Larsen and Yi Wang. Time-abstracted bisimulation: Implicit specifications and decidability. *Information and Computation*, 134(2):75 – 101, 1997.

20 Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.

21 Marino Miculan and Marco Peressotti. Structural operational semantics for non-deterministic processes with quantitative aspects. *Theoretical Computer Science*, 2016.

22 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

23 Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, 9(1), 2013.

24 Alexandra Silva and Bram Westerbaan. A coalgebraic view of ϵ-transitions. In Reiko Heckel and Stefan Milius, editors, *Proc. CALCO*, volume 8089 of *Lecture Notes in Computer Science*, pages 267–281. Springer, 2013.

25 Paweł Sobociński. Relational presheaves, change of base and weak simulation. *Journal of Computer and System Sciences*, 81(5):901–910, 2015.

26 Sam Staton. Relating coalgebraic notions of bisimulation. *Logical Methods in Computer Science*, 7(1), 2011.

27 Daniele Turi and Gordon Plotkin. Towards a mathematical operational semantics. In *Proc. LICS*, pages 280–291. IEEE Computer Society Press, 1997.