Explaining Non-Bisimilarity in a Coalgebraic Approach: Games and Distinguishing Formulas*

Barbara König[1], Christina Mika-Michalski[1], and Lutz Schröder[2]

[1] University of Duisburg-Essen, Germany
[2] Friedrich-Alexander-Universität Erlangen-Nŕnberg
{barbara_koenig,christina.mika-michalski}@uni-due.de,
lutz.schroeder@fau.de

Abstract. Behavioural equivalences can be characterized via bisimulation, modal logics, and spoiler-duplicator games. In this paper we work in the general setting of coalgebra and focus on generic algorithms for computing the winning strategies of both players in a bisimulation game. The winning strategy of the spoiler (if it exists) is then transformed into a modal formula that distinguishes the given non-bisimilar states. The modalities required for the formula are also synthesized on-the-fly, and we present a recipe for re-coding the formula with different modalities, given a separating set of predicate liftings. Both the game and the generation of the distinguishing formulas have been implemented in a tool called T-Beg.

Keywords: coalgebra · bisimulation game · distinguishing formulas · generic partition refinement

1 Introduction

There are many contexts in which it is useful to check whether two system states are behaviourally equivalent respectively bisimilar. In this way one can compare a system with its specification, replace a subsystem by another one that is behaviourally equivalent or minimize a transition system. Here we will concentrate on methods for explaining that two given states in a transition system are *not* bisimilar. The idea is to provide a witness that guarantees non-bisimilarity. Such a witness can be used to explain (to the user) why an implementation does not conform to a specification and give further insights for adjusting it.

The standard definition says that two states are bisimilar if they are related by a bisimulation relation. But this does not provide us with a immediate witness for non-bisimilarity, since we would have to enumerate all relations including that particular pair of states and show that they are not bisimulations. Hence, we have to resort to other characterizations of bisimilarity: bisimulation games [19], also known as spoiler-duplicator games, and modal logic. In the former case a proof of the non-bisimilarity of two states is given by a winning strategy of

* Supported by the project BEMEGA funded by the DFG.

the spoiler. In the latter case the Hennessy-Milner theorem [10] guarantees for image-finite labelled transition systems that, given two non-bisimilar states x, y, there exists a modal logic formula φ such that one of the states satisfies φ and the other does not. The computation of such distinguishing formulas is explained in [5].

While the results and techniques above have been introduced for labelled transition systems, we are here interested in the more general setting of coalgebras [17], which encompass various types of transition systems. Here we concentrate on coalgebras living in **Set**, where an endofunctor $F\colon \mathbf{Set} \to \mathbf{Set}$ specifies the branching type of the coalgebra (non-deterministic, probabilistic, etc.).

Modal logics have been extensively studied for coalgebras and it has been shown that under certain restrictions, modal coalgebraic logic is expressive, i.e., it satisfies the Hennessy-Milner theorem [16, 18]. However, to our knowledge, no explicit construction of distinguishing formulas has yet been given.

Games have been studied to a lesser extent, we mainly refer to [2], where the game is based on providing partial bisimulation relations, our own contribution [14], on which this article is based, and [12], which considers games from an abstract, fibrational perspective.

We combine both the game and the logic view, by first describing how to compute the winning strategies of the players. Then we show how one can construct a distinguishing formula based on the spoiler strategy. The modalities for the formula are not provided a priori, but are synthesized on-the-fly as so-called *cone modalities* while generating the formula. Furthermore we show under which conditions one can re-code a formula with such modalities into a formula with different modalities, given by a separating set of predicate liftings.

Both the game and the generation of the distinguishing formulas have been implemented in a generic tool called T-Beg[3], where the functor is provided as a parameter. In particular, using this tool, one can visualize coalgebras, play the game (against the computer), derive winning strategies and convert the winning strategy of the spoiler into a distinguishing formula.

Since the development of the tool was our central aim, we have made design decisions in such a way that we obtain effective algorithms. This means that we have taken a hands-on approach and avoided constructions that potentially iterate over infinitely many elements (such as the set of all modalities, which might be infinite).

The partition refinement algorithm presented in the paper distinguishes states that are not behaviourally equivalent by a single equivalence class, similar to known algorithms for checking bisimilarity in labelled transition systems [11, 15]. This requires a certain assumption on the endofunctor specifying the branching type (dubbed *separability by singletons*). In particular [15] has already been generalized to a coalgebraic setting in [6], using the assumption of *zippability*. Here we compare these two assumptions.

After presenting the preliminaries (Section 2), including the game, we describe how to compute the winning strategies in Section 3. In Section 4 we

[3] `Available at: https://www.uni-due.de/theoinf/research/tools_tbeg.php`

describe how to construct and re-code distinguishing formulas, followed by a presentation of the tool T-Beg in Section 5. Finally, we conclude in Section 6. The proofs can be found in Appendix A.

2 Preliminaries

Equivalence relations and characteristic functions: Let $R \subseteq X \times X$ be an *equivalence relation*, where the set of all equivalence relations on X is given by $Eq(X)$. For $x \in R$ we denote the *equivalence class* of x by $[x]_R = \{y \in X \mid (x, y) \in R\}$. By $E(R)$ we denote the set of all equivalence classes of R. Given $Y \subseteq X$, we define the *R-closure* of Y as follows: $[Y]_R = \{y \in X \mid \exists x \in X\,(x, y) \in R\}$.

For $Y \subseteq X$, we denote its *predicate* or *characteristic function* by $\chi_Y \colon X \to \{0, 1\}$. Furthermore, given a characteristic function $\chi \colon X \to \{0, 1\}$, its corresponding set is denoted $\hat{\chi} \subseteq X$.

We will sometimes overload the notation and for instance write $[p]_R$ for the R-closure of a predicate p. Furthermore we will write $p_1 \cap p_2$ for the intersection of two predicates.

Coalgebra: We restrict our setting to the category **Set**, in particular we assume an *endofunctor* $F \colon \mathbf{Set} \to \mathbf{Set}$, intuitively describing the branching type of the transition system under consideration. We assume that F preserves weak pullbacks.

A *coalgebra* [17], describing a transition system of this branching type, is given by a function $\alpha \colon X \to FX$. Two states $x, y \in X$ are *behaviourally equivalent* ($x \sim y$) if there exists a coalgebra homomorphism f from α to some coalgebra $\beta \colon Y \to FY$ (i.e., a function $f \colon X \to Y$ with $\beta \circ f = Ff \circ \alpha$) such that $f(x) = f(y)$.

Preorder lifting: Furthermore we need to *lift* preorders under a functor F. To this end, we use the lifting introduced in [1] (essentially the standard *Barr extension* of F [3, 20]), which guarantees that the lifted relation is again a preorder provided that F preserves weak pullbacks: Let $\leq$ be a preorder on Y, i.e. $\leq \subseteq Y \times Y$. We define the preorder $\leq^F \subseteq FY \times FY$ as follows: given $t_1, t_2 \in FY$, it holds that $t_1 \leq^F t_2$ whenever there exists some $t \in F(\leq)$ such that $F\pi_i(t) = t_i$, where $\pi_i \colon {\leq} \to Y$ with $i \in \{1, 2\}$ are the usual projections. More concretely, we consider in this paper the order $\leq = \{(0, 0), (0, 1), (1, 1)\}$ over $2 = \{0, 1\}$ and its corresponding liftings $\leq^F$.

Note that applying the functor is monotone wrt. the lifted order:

Lemma 1 ([14]). *Let $(Y, \leq)$ be an ordered set and let $p_1, p_2 \colon X \to Y$ be two functions. Then $p_1 \leq p_2$ implies $Fp_1 \leq^F Fp_2$ (with both inequalities read pointwise).*

Predicate liftings: In order to define the modal logic, we need the notion of *predicate liftings* (also called *modalities*). Formally, a predicate lifting for F is

a natural transformation $\lambda\colon \hat{\mathcal{P}} \Rightarrow \hat{\mathcal{P}}F$, where $\hat{\mathcal{P}}$ is the contravariant powerset functor. It transforms every subset $P \subseteq X$ into $\lambda(P) \subseteq FX$.

In this paper we use the fact that predicate liftings are in one-to-one correspondence with functions of type $\lambda\colon F2 \to 2$ (which specify subsets of $F2$ and will also be called *evaluation maps*) [18]. We view subsets $P \subseteq X$ as predicates $p = \chi_P$ and lift them via $p \mapsto \lambda \circ Fp$. In order to obtain expressive logics, we also need the notion of a separating set of predicate liftings.

Definition 2. *A set Λ of evaluation maps is separating for a functor $F\colon \boldsymbol{Set} \to \boldsymbol{Set}$ whenever for all sets X and $t_1, t_2 \in FX$ with $t_1 \neq t_2$ there exists $\lambda \in \Lambda$ and $p\colon X \to 2$ such that $\lambda(Fp(t_1)) \neq \lambda(Fp(t_2))$.*

This means that every $t \in FX$ is uniquely determined by the set $\{(\lambda, p) \mid \lambda \in \Lambda, p\colon X \to 2, \lambda(Fp(t)) = 1\}$. Such a separating set of predicate liftings exists iff $(Fp\colon FX \to F2)_{p\colon X\to 2}$ is jointly injective.

Here we concentrate on *unary* predicate liftings: If one generalizes to *polyadic* predicate liftings, separation can be shown for every accessible functor [18].

Separating sets of monotone predicate liftings and the lifted order on $F2$ are related as follows:

Proposition 3 ([14]). *Let $\lambda\colon F2 \to 2$ be an evaluation map, mapping to $(2, \leq)$. It induces a monotone predicate lifting $(p\colon X \to 2) \mapsto (\lambda \circ Fp\colon FX \to 2)$ iff $\lambda\colon (F2, \leq^F) \to (2, \leq)$ is monotone.*

Proposition 4 ([14]). *F has a separating set of monotone evaluation maps iff $\leq^F \subseteq F2 \times F2$ is anti-symmetric and $(Fp\colon FX \to F2)_{p\colon X\to 2}$ is jointly injective.*

Coalgebraic modal logics: Given a cardinal κ and a set Λ of evaluation maps $\lambda\colon F2 \to 2$, we define a coalgebraic modal logic $\mathcal{L}^\kappa(\Lambda)$ via the grammar:

$$\varphi ::= \bigwedge \Phi \mid \neg\varphi \mid [\lambda]\varphi \quad \text{where } \Phi \subseteq \mathcal{L}^\kappa(\Lambda) \text{ with } \mathit{card}(\Phi) < \kappa \text{ and } \lambda \in \Lambda.$$

The last case describes the prefixing of a formula φ with a modality $[\lambda]$. Given a coalgebra $\alpha\colon X \to FX$ and a formula φ, the semantics of such a formula is given by a map $[\![\varphi]\!]_\alpha\colon X \to 2$, where conjunction and negation are interpreted as usual and $[\![[\lambda]\varphi]\!]_\alpha = \lambda \circ F[\![\varphi]\!]_\alpha \circ \alpha$.

For simplicity we will often write $[\![\varphi]\!]$ instead of $[\![\varphi]\!]_\alpha$. Furthermore for $x \in X$, we write $x \models \varphi$ whenever $[\![\varphi]\!](x) = 1$. As usual, whenever $[\![\varphi]\!]_\alpha = [\![\psi]\!]_\alpha$ for all coalgebras α we write $\varphi \equiv \psi$. We will use derived operators such as tt (empty conjunction), ff ($\neg tt$) and $\bigvee$ (disjunction).

The logic is always sound, i.e., two behaviourally equivalent states satisfy the same formulas. Furthermore whenever F is κ-accessible and the set Λ of predicate liftings is separating, it can be shown that the logic is also expressive, i.e., two states that satisfy the same formulas are behaviourally equivalent.

Game characterizing behavioural equivalence: We will now present the rules for a *behavioural equivalence game*, see [14]. At the beginning of a game, two states x, y are given. The aim of the spoiler (S) is to prove that $x \nsim y$, the duplicator (D) attempts to show the opposite.

- **Initial situation:** A coalgebra $\alpha\colon X \to FX$ and two states $x, y \in X$.
- **Step 1:** S chooses $s \in \{x, y\}$ and a predicate $p_1\colon X \to 2$.
- **Step 2:** D takes $t \in \{x, y\}\backslash\{s\}$ if $x \neq y$ and $t = s$ otherwise and has to answer with a predicate $p_2\colon X \to 2$ satisfying $Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t))$.
- **Step 3:** S chooses p_i with $i \in \{1, 2\}$ and some state $x' \in X$ with $p_i(x') = 1$.
- **Step 4:** D chooses some state $y' \in X$ with $p_j(y') = 1$ where $i \neq j$.

After one round the game continues in Step 1 with the pair (x', y') of states in the case $i = 1$ and with (y', x') if $i = 2$. D wins if the game continues forever or if S has no move at Step 3. In all other cases, i.e. D has no move at Step 2 or Step 4, S wins.

The game characterizes behavioural equivalence for functors that are weak pullback preserving and for which the lifted order $\leq^F$ is anti-symmetric. Then it holds that $x \sim y$ if and only if D has a winning strategy from the initial situation (x, y). As already shown in [14], in the case of two non-bisimilar states $x \nsim y$ we can convert a modal logic formula φ distinguishing x, y, i.e., $x \models \varphi$ and $y \not\models \varphi$, into a winning strategy for the spoiler. Furthermore we can extract the winning strategy for the duplicator from the bisimulation relation.

However, in [14] we did not yet show how to directly derive the winning strategy of both players or how to construct a distinguishing formula φ.

3 Computation of Winning Strategies

In the rest of the paper we will fix a coalgebra $\alpha : X \to FX$ with finite X for a weak pullback preserving endofunctor $F\colon \mathbf{Set} \to \mathbf{Set}$. Furthermore we assume that F has a separating set of monotone predicate liftings, which implies that $\leq^F$, the lifted order on 2, is anti-symmetric, hence a partial order.

We first present a simple but generic algorithm to derive the winning strategy for the spoiler (S) and duplicator (D) for a given coalgebra $\alpha\colon X \to FX$. This is based on a fixpoint iteration that determines those pairs of states $(x, y) \in X \times X$ for which D has a winning strategy, i.e., those pairs of states that are behaviourally equivalent. In particular we consider the greatest fixpoint of the following monotone function $\mathcal{F}_\alpha\colon Eq(X) \to Eq(X)$ on equivalence relations:

$$\mathcal{F}_\alpha(R) = \{(x, y) \in R \mid \forall P \in E(R)\colon F\chi_P(\alpha(x)) = F\chi_P(\alpha(y))\}$$

As shown in [14], the relation

$$W_\alpha = \{(x, y) \in X \times X \mid \text{there exists a winning strategy of } D \text{ for } (x, y)\}$$

coincides with behavioural equivalence $\sim$.

We will in the following prove that the greatest fixpoint of $\mathcal{F}_\alpha$ (i.e. $\nu\mathcal{F}_\alpha$) coincides with W_α and hence gives us bisimilarity. One direction of the proof deals with deriving a winning strategy for S for each pair $(x, y) \notin \nu\mathcal{F}_\alpha$. In order to explicitly extract such a winning strategy for S – which will also be important later when we construct the distinguishing formula – we will slightly adapt the algorithm based on fixpoint iteration. Before we explain this, we formally define and explain the strategies of D and S.

We start with the winning strategy of the duplicator in the case where the two given states are bisimilar. This strategy has already been presented in [14], but we describe it here again explicitly. The duplicator only has to know a suitable coalgebra homomorphism.

Proposition 5 (Strategy of the duplicator, [14]). *Let $\alpha\colon X \to FX$ be a coalgebra. Assume that D, S play the game on an initial situation (x, y) with $x \sim y$. This means that there exists a coalgebra homomorphism $f\colon X \to Z$ from α to a coalgebra $\beta\colon Z \to FZ$ such that $f(x) = f(y)$.*

Assume that in Step 2 D answers with $p_2 = [p_1]_{ker(f)}$, i.e., p_2 is the $ker(f)$-closure[4] of the predicate p_1. (In other words: $p_2(x) = 1$ iff there exists $y \in X$ such that $f(x) = f(y)$ and $p_1(y) = 1$.)

Then the condition of Step 2 is satisfied and in Step 4 D is always able to pick a state y' with $p_j(y') = 1$ and $f(x') = f(y')$.

We give a short summary why this winning strategy is valid: since f is a coalgebra homomorphism we have $Ff(\alpha(x)) = \beta(f(x)) = \beta(f(y)) = Ff(\alpha(y))$. By construction p_2 factors through f, that is $p_2 = p_1' \circ f$ for some $p_1'\colon Z \to 2$. This implies $Fp_2(\alpha(x)) = Fp_1'(Ff(\alpha(x))) = Fp_1'(Ff(\alpha(y))) = Fp_2(\alpha(y))$. Since $p_1 \leq p_2$ it follows from monotonicity (Lemma 1) that $Fp_1(\alpha(x)) \leq^F Fp_2(\alpha(x)) = Fp_2(\alpha(y))$. Hence p_2 satisfies the conditions of Step 2. Furthermore if the spoiler picks a state x' in p_1 in Step 3, the duplicator can pick the same state in p_2 in Step 4. If instead the spoiler picks a state x' in p_2, the duplicator can at least pick a state y' in p_1 which satisfies $f(x') = f(y')$, which means that the game can continue.

We now switch to the spoiler strategy that can be used to explain why the states are not behaviourally equivalent.

Definition 6 (Strategy of the spoiler). *A strategy for the spoiler is given by the following pair of functions:*

$$I\colon X \times X \to \mathbb{N}_0 \cup \{\infty\} \text{ and } T\colon (X \times X)\backslash\nu\mathcal{F}_\alpha \to X \times \mathcal{P}X.$$

Here $I(x, y)$ denotes the first index where x, y are separated in the fixpoint iteration of $\mathcal{F}_\alpha$. More precisely, i is the least least index i such that $(x, y) \notin \mathcal{F}_\alpha^i(X \times X)$. If two states x, y are never separated, i.e., $(x, y) \in \nu\mathcal{F}_\alpha$ we set $I(x, y) = \infty$.

The second component T tells the spoiler what to play in Step 2. In particular whenever $T(x, y) = (s, P_1)$, S will play s and $p_1 = \chi_{P_1}$.

[4] For a function $f\colon X \to Y$, $ker(f) = \{(x_1, x_2) \mid x_1, x_2 \in X, f(x_1) = f(y_2)\} \subseteq X \times X$.

Such a winning strategy for the spoiler can be computed during the fixpoint iteration, see Algorithm 1. Assume that the algorithm terminates after n steps and returns R_n. It is easy to see that R_n coincides with $\nu\mathcal{F}_\alpha$: as usual for a partition refinement algorithm, we start with the coarsest relation $R_0 = X \times X$. Since $\leq^F$ is, by assumption, anti-symmetric $F\chi_{P_1}(\alpha(x)) \leq^F F\chi_{P_1}(\alpha(y))$ and $F\chi_{P_1}(\alpha(y)) \leq^F F\chi_{P_1}(\alpha(x))$ are equivalent to $F\chi_{P_1}(\alpha(x)) = F\chi_{P_1}(\alpha(y))$ and the algorithm removes a pair (x, y) from the relation iff this condition does not hold.

Every relation R_i is finer than its predecessor R_{i-1} and, since $\mathcal{F}_\alpha$ preserves equivalences, each is an equivalence relation. Since we are assuming a finite set X of states, the algorithm will eventually terminate.

Algorithm 1 Computation of $\nu\mathcal{F}_\alpha$ and the winning strategy of the spoiler

procedure COMPUTE GREATEST FIXPOINT OF $\mathcal{F}_\alpha$ AND WINNING MOVES FOR S
  for all $(x, y) \in X \times X$ **do**
    $I(x, y) \leftarrow \infty$
  $i \leftarrow 0$, $R_0 \leftarrow X \times X$
  repeat
    $i \leftarrow i + 1$, $R_i \leftarrow R_{i-1}$
    for all $(x, y) \in R_{i-1}$ **do**
      for all $P \in E(R_{i-1})$ **do**
        if $F\chi_P(\alpha(x)) \not\leq^F F\chi_P(\alpha(y))$ **then**
          $T(x, y) \leftarrow (x, P)$, $I(x, y) \leftarrow i$, $R_i \leftarrow R_i \setminus \{(x, y)\}$
        else
          if $F\chi_P(\alpha(y)) \not\leq^F F\chi_P(\alpha(x))$ **then**
            $T(x, y) \leftarrow (y, P)$, $I(x, y) \leftarrow i$, $R_i \leftarrow R_i \setminus \{(x, y)\}$
  until $R_{i-1} = R_i$
return R_i, T, I

In addition, $T(x, y)$ and $I(x, y)$ are updated in every step, where we distinguish whether $Fp(\alpha(x)) \not\leq^F Fp(\alpha(y))$ or the other inequality hold. We will now show that Algorithm 1 indeed computes a winning strategy for the spoiler.

Proposition 7. *Assume that $R_n = \nu\mathcal{F}_\alpha, T, I$ have been computed by Algorithm 1. Furthermore let $(x, y) \notin R_n$, which means that $I(x, y) < \infty$ and $T(x, y)$ is defined. Then the following constitutes a winning strategy for the spoiler:*

- *Let $T(x, y) = (s, P_1)$. Then in Step 1 S plays a predicate $p_1 = \chi_{P_1}$ and $s \in \{x, y\}$.*
- *Assume that in Step 2 D answers with a state t and a predicate p_2 such that $Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t))$.*
- *Then, in Step 3 there exists a state $y' \in X$ such that $p_2(y') = 1$ and $I(x', y') < I(x, y)$ for all $x' \in X$ with $p_1(x') = 1$. S will hence select p_2 and this state y'.*
- *Next, in Step 4 D selects some x' with $p_1(x') = 1$ and the game continues with $(x', y') \notin R_n$.*

Finally we show that $\nu\mathcal{F}_\alpha$ coincides with W_α and therefore also with behavioural equivalence $\sim$ (see [14]). For this we need one further requirement on the functor:

Definition 8. *Let $F\colon \boldsymbol{Set} \to \boldsymbol{Set}$ be an endofunctor on $\boldsymbol{Set}$. We say that F is* separable by singletons for a set X *if the following holds: for all $t_1 \neq t_2$ with $t_1, t_2 \in FX$, there exists $p\colon X \to 2$ where $p(x) = 1$ for exactly one $x \in X$ (i.e., p is a singleton) and $Fp(t_1) \neq Fp(t_2)$.*

It is separable by singletons *if it is separable by singletons for all sets X.*

It is obvious that separability by singletons implies the existence of a separable set of predicate liftings, however the reverse implication does not hold as the following example shows.

Example 9. In this paper we will frequently consider the finite powerset functor $\mathcal{P}_f$ and the finitely supported probability distribution functor $\mathcal{D}$ (which are both ω-accessible and hence yield a logic with only finite formulas.) In addition, both are separable by singletons.

A functor that does not have this property, but does have a separating set of predicate liftings, is the neighbourhood functor $\mathcal{N} = \check{\mathcal{P}}\check{\mathcal{P}}$ [7], the composition of the contravariant powerset functor with itself. Consider $X = \{a, b, c, d\}$ and two elements $t_1, t_2 \in \mathcal{N}X$ where $t_1 = \{\{a, b\}\}$, $t_2 = \{\{c, d\}\}$. For any singleton predicate p the image of $\check{\mathcal{P}}p\colon \mathcal{P}2 \to \mathcal{P}X$ does not contain a two-element set, hence $\check{\mathcal{P}}\check{\mathcal{P}}p(t_1) = \emptyset = \check{\mathcal{P}}\check{\mathcal{P}}p(t_2)$ and t_1, t_2 can not be distinguished.

However, this functor is ruled out anyway in our setting, since it does not preserve weak pullbacks.

We are now ready to prove the following theorem.

Theorem 10. *Let $\alpha : X \to FX$ be a coalgebra where F is separable by singletons. It holds that $\nu\mathcal{F}_\alpha = W_\alpha$, i.e., it contains exactly the pairs $(x, y) \in X \times X$ for which the duplicator has a winning strategy.*

We now give an example for the application of the algorithm that also illustrates the differences between our generic game and the classical bisimulation game for labelled transition systems [19].

Example 11. Consider the transition system in Figure 1, which depicts a coalgebra $\alpha\colon X \to \mathcal{P}_f(A \times X)$. Clearly $x \nsim y$.

First consider the classical game where one possible winning strategy of the spoiler is as follows: he moves $x = 1 \xrightarrow{a} 4$, which must be answered by the duplicator via $y = 2 \xrightarrow{a} 5$. Now the spoiler switches and makes a move $5 \xrightarrow{a} 8$, which can not be answered by the duplicator.

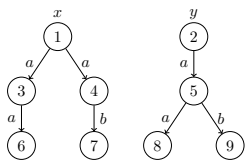

Fig. 1: Spoiler has a winning strategy at (x, y).

In our case the corresponding game proceeds as follows: the spoiler chooses $s = x$ and $p_1 = \chi_{\{4\}}$. Now

the duplicator takes $t = y$ and can for instance answer with $p_2 = \chi_{\{5\}}$, which leads to

$$Fp_1(\alpha(x)) = \{(a,0),(a,1)\} \leq^F \{(a,1)\} = Fp_2(\alpha(y))$$

Regardless of how S and D choose states, the next game situation is $(4,5)$.

Now the spoiler is *not* forced to switch, but can choose $s = 4$ and can play basically any predicate p_1, which leads to either $Fp_1(\alpha(4)) = \{(b,1)\}$ or $Fp_1(\alpha(4)) = \{(b,0)\}$. D has no answering move, since $Fp_2(\alpha(5))$ will always contain tuples with b *and* c, which are not in $\leq^F$-relation with the move of S (see also Figure 2, which depicts $F2$ and its order).

If we execute Algorithm 1, we first use the predicate χ_X to separate $\{1,2,3\}$ (with value $\{(a,1)\}$) from $\{4\}$ (with value $\{(b,1)\}$), $\{5\}$ (with value $\{(a,1),(b,1)\}$ and $\{6,7,8,9\}$ (with value $\emptyset$). In the second step the predicate $\chi_{\{4\}}$ is employed to separate $\{1\}$ (with value $\{(a,0),(a,1)\}$) from $\{2\}$(with value $\{(a,0)\}$). Similarly $\{3\}$ can be separated from both $\{1\}$ and $\{2\}$ with the predicate $\chi_{\{6,7,8,9\}}$. This also determines the strategy of the spoiler explained above.

The notion of separability by singletons is needed, since the partition refinement algorithm which we are using separates two states based on a *single* equivalence class of their successors, whereas other partition refinement algorithms such as [13] consider *all* equivalence classes. As shown in Example 9 this is indeed a restriction, however such additional assumptions seem necessary if we want to adapt efficient bisimulation checking algorithms such as the ones by Kanellakis/Smolka [11] or Paige/Tarjan [15] to the coalgebraic setting. In fact, the Paige/Tarjan algorithm already has a coalgebraic version [6] which operates under the assumption that the functor is *zippable*. Here we show that the related notion of m-zippability is very similar to separability. (The zippability of [6] is in fact 2-zippability.)

Definition 12 (zippability). *A functor F is* m-zippable *if for arbitrary sets $A_1, \ldots, A_m$ the following morphism is injective:*

$$F(A_1 + \cdots + A_m) \xrightarrow{\langle F(f_1),\ldots,F(f_m)\rangle} F(A_1 + 1) \times \cdots \times F(A_m + 1)$$

where $f_i = id_{A_i} + !$, with $! \colon A_1 + \cdots + A_{i-1} + A_{i+1} + \cdots + A_m \to 1$, is the function mapping all elements of A_i to themselves and all other elements to $\bullet$ (assuming that $1 = \{\bullet\}$).

Lemma 13. *Assume that $F \colon \mathbf{Set} \to \mathbf{Set}$ is a functor preserving injections. We have that if F is separable by singletons, it is m-zippable. Furthermore if F is m-zippable, then it is separable by singletons for all sets X with $|X| \leq m$.*

Runtime Analysis If X is finite our algorithm always terminates and has a polynomial runtime. If $|X| = n$, the algorithm makes at most n iterations, since there can be at most n splits of equivalence classes. In each iteration we consider

up to n^2 pairs of states and in order to decide whether a pair can be separated, we have to consider up to n equivalence classes.

Hence a naive implementation of the algorithm has a runtime of $O(n^4)$. We expect however that there are optimizations based on the algorithms by Kanellakis and Smolka [11] and Paige and Tarjan [15]. In particular, it would be interesting to incorporate the generalization of the Paige-Tarjan algorithm to the coalgebraic setting [6].

4 Construction of Distinguishing Formulas

Next we illustrate how to derive a distinguishing modal logic formula from the winning strategy of S computed by Algorithm 1. The other direction (obtaining the winning strategy from a distinguishing formula) has been covered in [14].

4.1 Cone Modalities

We focus on an on-the-fly extraction of relevant modalities, to our knowledge a new contribution, and discuss the connection to the minimal set of separating predicate liftings.

One way of enabling the construction of formulas is to specify the separating set of predicate liftings Λ in advance. But this set might be infinite and hard to represent. Instead here we generate the modalities while constructing the formula. We focus in particular on so-called *cone modalities*: given $v \in F2$ we take the upward-closure of v as a modality. We also explain how logical formulas with cone modalities can be translated into other separating sets of modalities.

Definition 14 (Cone modalities). *Let $v \in F2$. A cone modality $[\uparrow v]$ is given by the following evaluation map $\uparrow v : F2 \to 2$:*

$$\uparrow v(u) = \lambda(u) = \begin{cases} 1, & \text{if } v \leq^F u \\ 0, & \text{otherwise} \end{cases}$$

Obviously these evaluation maps yield a separating set of predicate liftings (provided that the functor has such a set): if $v_1 \neq v_2$ (for $v_1, v_2 \in F2$), either $v_1 \not\leq^F v_2$ or $v_2 \not\leq^F v_1$, since we require that the lifted order is anti-symmetric on $F2$. In the first case $\uparrow v_1(v_1) = 1$ and $\uparrow v_1(v_2) = 0$, in the second case $\uparrow v_2$ is the distinguishing evaluation map.

Example 15. We will discuss modalities respectively evaluation maps in more detail for the functor $FX = \mathcal{P}_f(A \times X)$, which specifies the branching type of labelled transition systems. In our example $A = \{a, b\}$. The set $F2$ with order $\leq^F$ is depicted as a Hasse diagram in Figure 2. For every element there is a cone modality, 16 modalities in total. It is known from the Hennessy-Milner theorem [10] that two modalities are enough: either $\Box_a, \Box_b$ (box modalities) or $\Diamond_a, \Diamond_b$ (diamond modalities), where for $v \in F2$:

In Figure 2 $\Box_a$ respectively $\Diamond_a$ are represented by the elements above the two lines (solid respectively dashed).

$$\Box_a(v) = \begin{cases} 1 & \text{if } v \cap \{(a,0)\} = \emptyset \\ 0 & \text{otherwise} \end{cases} \qquad \Diamond_a(v) = \begin{cases} 1 & \text{if } (a,1) \in v \\ 0 & \text{otherwise} \end{cases}$$

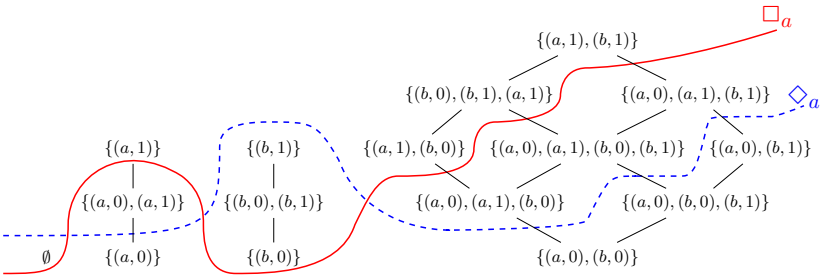

Fig. 2: Set $F2$ with order $\leq^F$ (for labelled transition systems). $\Box_a$ and $\Diamond_a$ are given by all values above the drawn (dashed) lines.

Example 16. As a second example we discuss the functor $FX = (\mathcal{D}X + 1)^A$, specifying probabilistic transition systems. The singleton set $1 = \{\bullet\}$ denotes termination. Again we set $A = \{a, b\}$.

Note that since $\mathcal{D}2$ is isomorphic to the interval $[0, 1]$, we can simply represent any distribution $d\colon 2 \to [0, 1]$ by $d(1)$. Hence $F2 \cong ([0, 1]+1)^A$. The partial order is obtained pointwise and is depicted in Figure 3: it decomposes into four disjoint partial orders, depending on whether both a, b, neither or one of them is mapped to $\bullet$. The right-hand part of this partial order consists of function $[0, 1]^A$ with the pointwise order. We will also abbreviate a map $[a \mapsto p, b \mapsto q]$ by $\langle a_p, b_q \rangle$.

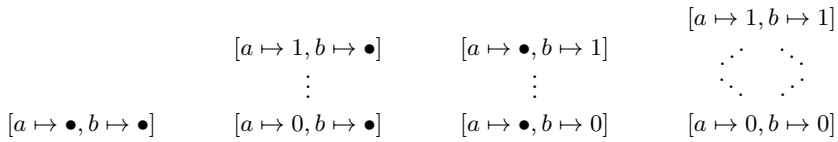

Fig. 3: $F2$ with order $\leq^F$ (for probabilistic transition systems).

4.2 From Winning Strategies to Distinguishing Formulas

We will now show how a winning strategy of S can be transformed into a distinguishing formula, based on cone modalities, including some examples.

The basic idea behind the construction is the following: Let (x, y) be a pair of states separated during Step i of the partition refinement algorithm (Algorithm 1). This means hat we have the following situation: $F\chi_P(\alpha(x)) \not\leq^F F\chi_P(\alpha(y))$ (or vice versa) for some equivalence class P of R_{i-1}. Based on

$v = F\chi_P(\alpha(x))$ we define a cone modality $\lambda = {\uparrow} v$. Now assume that if we can characterize P by some formula ψ, i.e., $[\![\psi]\!] = \chi_P$ (we will later show that this is always possible), we can define the formula $\varphi = [\lambda]\psi$. Then it holds that:

$$[\![\varphi]\!](x) = \lambda(F[\![\psi]\!](\alpha(x))) = {\uparrow} v(F\chi_P(\alpha(x))) = 1$$
$$[\![\varphi]\!](y) = \lambda(F[\![\psi]\!](\alpha(y))) = {\uparrow} v(F\chi_P(\alpha(y))) = 0$$

That is we have $x \models \varphi$ and $y \not\models \varphi$, which means that we have constructed a distinguishing formula for x, y.

First, we describe how a winning strategy for the spoiler for a pair (x, y) is converted into a formula and then prove that this formula distinguishes x, y.

Definition 17. *Let $x \nsim y$ (equivalently $(x, y) \notin R_n$) and let (T, I) be the winning strategy for the spoiler computed by Algorithm 1. We construct a formula $\varphi_{x,y}$ as follows: assume that $T(x, y) = (s, P)$ where $s = x$. Then set $v = F\chi_P(\alpha(x))$, $\lambda = {\uparrow} v$ and define $\varphi_{x,y} = [\lambda]\varphi$, where φ is constructed in the following way:*

- $I(x, y) = 1$: $\varphi = tt$
- $I(x, y) > 1$: $\varphi = \bigvee_{x' \in P} \big(\bigwedge_{y' \in X \setminus P} \varphi_{x',y'} \big)$

Whenever we have $s = y$ we instead define $v = F\chi_P(\alpha(y))$ and $\varphi_{x,y} = \neg[\lambda]\varphi$.

This encoding is well-defined, because it always holds that $I(x', y') < I(x, y)$ (since P is an equivalence class of R_i where $i = I(x, y)$).

Proposition 18. *Let $\alpha : X \to FX$ be a coalgebra and assume that we have computed R_n, T, I with Algorithm 1. Then, given $(x, y) \notin R_n$, the construction in Definition 17 yields a formula $\varphi_{x,y} \in \mathcal{L}^\kappa(\Lambda)$ such that $x \vDash \varphi_{x,y}$ and $y \nvDash \varphi_{x,y}$.*

We next present an optimization of the construction in Definition 17, inspired by [5]. In the case $I(x, y) > 1$ one can pick an arbitrary $x' \in P$ and keep only one element of the disjunction. In order to show that this simplification is permissible, we need the following lemma.

Lemma 19. *Given two states $(x, y) \notin R_n$ and a distinguishing formula $\varphi_{x,y}$ based on Definition 17. Let (x', y') be given such that $I(x', y') > I(x, y)$. Then $x' \vDash \varphi_{x,y}$ if and only if $y' \vDash \varphi_{x,y}$.*

Now we can show that we can replace the formula φ from 17 by a simpler formula φ'.

Lemma 20. *Let $(x, y) \notin R_i$ and let P be an equivalence class of R_{i-1}. Furthermore let*

$$\varphi' = \bigwedge_{y' \in X \setminus P} \varphi_{x',y'}$$

for some $x' \in P$. Then $[\![\varphi']\!] = \chi_P$.

Finally, we can simplify our construction described in Definition 17 to only one inner conjunction.

Corollary 21. *We use the construction of $\varphi_{x,y}$ as described in Definition 17 with the only modification that for $I(x,y) > 1$ the formula φ is replaced by the formula φ' from Lemma 19 for some $x' \in P$. Then this yields a formula $\varphi_{x,y}$ such that $x \vDash \varphi_{x,y}$ and $y \nvDash \varphi_{x,y}$.*

A further optimization takes only one representative y' from every equivalence class different from P.

Before we give an overview of the tool T-Beg that generates such formulas in the next section, we explore two slightly more complicated examples.

Example 22. Take the coalgebra for the functor $FX = (\mathcal{D}X + 1)^A$ depicted in Figure 4. Note that $A = \{a, b\}$ and $X = \{1, \dots, 5\}$. We have for instance $\alpha(3) = [a \mapsto \delta_3, b \mapsto \bullet]$ where δ_3 is the Dirac distribution. This is visualized by drawing an arrow labelled $a, 1$ from 3 to 3 and omitting b-labelled arrows.

We explain only selected steps of the construction: in the first step the partition refinement algorithm (Algorithm 1) separates 1 from 3 (among other separations), where the spoiler strategy is given by $T(1,3) = (1, X)$. In order to obtain a distinguishing formula we determine $v = F\chi_X(\alpha(1)) = \langle a_1, b_1\rangle$ (using the abbreviations explained in Example 16) and obtain $\varphi_{1,3} = [\uparrow\langle a_1, b_1\rangle]tt$. In fact, this formula also distinguishes 1 from 4, hence $\varphi_{1,3} = \varphi_{1,4}$. If, on the other hand, we would like to distinguish $3, 4$, we would obtain $\varphi_{3,4} = [\uparrow\langle a_1, b_\bullet\rangle]tt$.

After the first step, we obtain the partitions $\{1,2,5\}, \{3\}, \{4\}$. Now we consider states $1, 2$ which can be separated by playing $T(2,1) = (2, \{1,2,5\})$, since 5 behaves differently from 3. Again we compute $v = F\chi_P(\alpha(2)) = \langle a_{0.7}, b_{0.8}\rangle$ (for $P = \{1,2,5\}$) and obtain $\varphi_{2,1} = [\uparrow\langle a_{0.7}, b_{0.8}\rangle](\varphi_{1,3} \wedge \varphi_{1,4})$. Here we picked 1 as the representative of its equivalence class.

In summary, we obtain the following formula that is satisfied by 2, but not by 1:

$$[\uparrow\langle a_{0.7}, b_{0.8}\rangle][\uparrow\langle a_1, b_1\rangle]tt.$$

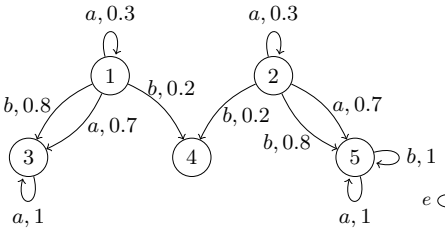

Fig. 4: Probabilistic transition system

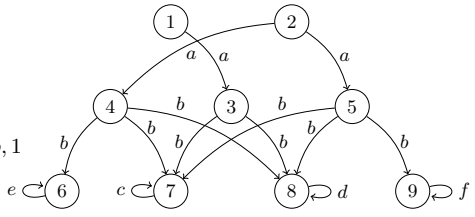

Fig. 5: Non deterministic transition system.

Example 23. We will now give an example where conjunction is required to obtain the distinguishing formula. We work with a coalgebra for the functor $FX = \mathcal{P}_f(A \times X)$, which is depicted in Figure 5. Note that $A = \{a, b, c, d, e, f\}$ and $X = \{1, \dots, 9\}$.

We explain only selected steps: in the first step the partition refinement separates 6 from 7 (among other separations), where the spoiler strategy is given by $T(6, 7) = (6, X)$. As explained above we determine $v = F\chi_X(\alpha(6)) = \{(e, 1)\}$ and obtain $\varphi_{6,7} = [\uparrow\{(e, 1)\}]tt$. In fact, this formula also distinguishes 6 from all other states, so we denote it by $\varphi_{6,*}$.

Next, we consider states $3, 4$, where the possible moves of 3 are a strict subset of the moves of 4. Hence the spoiler strategy is $T(3, 4) = (4, \{6\})$, i.e., the spoiler has to move to state 6 that is not reachable from 3. Again we compute $v = F\chi_P(\alpha(4)) = \{(b, 1), (b, 0)\}$ (for $P = \{6\}$) and obtain $\varphi_{3,4} = \neg[\uparrow\{(b, 1), (b, 0)\}]\varphi_{6,*}$. Note that this time we have to use negation, since the spoiler moves from the second state in the pair.

Finally we regard states $1, 2$ where the spoiler strategy is $T(1, 2) = (1, \{3\})$. We compute $v = F\chi_P(\alpha(1)) = \{(a, 1)\}$ (for $P = \{3\}$) and obtain $\varphi_{1,2} = [\uparrow\{(a, 1)\}]\big(\bigwedge_{x\in\{1,2,4,\dots,9\}} \varphi_{3,x}\big)$. In fact, here it is sufficient to consider $x = 4$ and $x = 5$, resulting in the following distinguishing formula:

$$[\uparrow\{(a, 1)\}]\big(\neg[\uparrow\{(b, 0), (b, 1)\}][\uparrow\{(e, 1)\}]tt \ \wedge \ \neg[\uparrow\{(b, 0), (b, 1)\}][\uparrow\{(f, 1)\}]tt\big).$$

4.3 Recoding Modalities

Finally, we will show under which conditions one can encode cone modalities into generic modalities, given by a separating set of predicate liftings Λ, not necessarily monotone. We first need the notion of strong separation.

Definition 24. *Let Λ be a separating set of predicate liftings of the form $\lambda\colon F2 \to 2$. We call Λ* strongly separating *if for every $t_1 \neq t_2$ with $t_1, t_2 \in F2$ there exists $\lambda \in \Lambda$ such that $\lambda(t_1) \neq \lambda(t_2)$.*

We can generate a set of strongly separating predicate liftings from every separating set of predicate liftings.

Lemma 25. *Let Λ be set of predicate liftings. Furthermore we denote the four functions on 2 by id_2, one (constant 1-function), zero (constant 0-function) and neg ($neg(0) = 1$, $neg(1) = 0$).*

Then

$$\Lambda' = \{\lambda, \lambda \circ Fone, \lambda \circ Fzero, \lambda \circ Fneg \mid \lambda \in \Lambda\}$$

is a set of strongly separating predicate liftings.

Furthermore for every formula φ we have that

$$[\lambda \circ Fone]\varphi \equiv [\lambda]tt \qquad [\lambda \circ Fzero]\varphi \equiv [\lambda]ff \qquad [\lambda \circ Fneg]\varphi \equiv [\lambda](\neg\varphi)$$

This means that we can still express the new modalities with the previous ones. Λ' is just an auxiliary construct that helps us to state the following proposition.

Proposition 26. *Assume that $F2$ is finite and let Λ be a strongly separating set of predicate liftings. Let furthermore $v \in F2$ and let φ be a formula.*

For a given $u \in F2$ we write $\Lambda_u = \{\lambda \in \Lambda \mid \lambda(u) = 1\}$. Then it holds that

$$[\uparrow v]\varphi \equiv \bigvee_{v \leq^F u} \Big(\bigwedge_{\lambda \in \Lambda_u} [\lambda]\varphi \wedge \bigwedge_{\lambda \notin \Lambda_u} \neg[\lambda]\varphi \Big)$$

By performing this encoding inductively, we can transform a formula with cone modalities into a formula with arbitrary modalities taken from Λ. The encoding preserves negation and conjunction, only the modalities are transformed.

Example 27. We come back to labelled transition systems and the functor $FX = \mathcal{P}_f(A \times X)$ with $A = \{a, b\}$. In this case the set $\{\Box_a, \Box_b, \Diamond_a, \Diamond_b\}$ of predicate liftings is strongly separating.

Now let $v = \{(a, 0), (b, 1)\} \in \mathcal{P}_f(A \times 2)$ and we show how to encode the corresponding cone modality using only box and diamond:

$$\begin{aligned}[\uparrow v]\varphi \equiv\ & (\neg\Box_a\varphi \wedge \Box_b\varphi \wedge \neg\Diamond_a\varphi \wedge \Diamond_b\varphi) \vee (\neg\Box_a\varphi \wedge \Box_b\varphi \wedge \Diamond_a\varphi \wedge \Diamond_b\varphi) \\ & \vee (\Box_a\varphi \wedge \Box_b\varphi \wedge \Diamond_a\varphi \wedge \Diamond_b\varphi)\end{aligned}$$

The first term describes $\{(a, 0), (b, 1)\}$, the second $\{(a, 0), (a, 1), (b, 1)\}$ and the third $\{(a, 1), (b, 1)\}$.

Note that we can not directly generalize Proposition 26 to the case where $F2$ is infinite. The reason for this is that the the disjunction over all $u \in F2$ such that $v \leq^F u$ might violate the cardinality constraints of the logic. Hence we will consider an alternative, where the re-coding works only under certain assumptions. We will start with the following example.

Example 28. Consider the functor $FX = (\mathcal{D}X + 1)^A$ (see also Example 16) and the corresponding (countable) separating set of (monotone) predicate liftings

$$\Lambda = \{\lambda_{(a,q)} \colon F2 \to 2 \mid a \in A, q \in [0, 1] \cap \mathbb{Q}\} \cup \{\lambda_{(a,\bullet)} \mid a \in A\}$$

where $\lambda_{(a,q)}(v) = 1$ if $v(a) \in \mathbb{R}$ and $v(a) \geq q$ and $\lambda_{(a,\bullet)} = 1$ if $v(a) = \bullet$. Here, a modality $[\lambda_{(a,q)}]$ indicates that the probability of making an a-transition is greater or equal than q and a modality $[\lambda_{(a,\bullet)}]$ tells us that we terminate with a.

The disjunction $\bigvee_{v \leq^F u}$ in the construction of $[\uparrow v]\varphi$ in Proposition 26 is in general uncountable and may hence fail to satisfy the cardinality constraints of the logic.

However, we can exploit certain properties of this set of predicate liftings, in order to re-code modalities.

Lemma 29. *Let F be the functor with $FX = (\mathcal{D}X + 1)^A$ and let Λ be the separating set of predicate liftings from Example 28. Furthermore let $v \in F2$. Then it holds that:*

$$\uparrow v = \bigcap_{\lambda \in \Lambda, \lambda(v) = 1} \lambda$$

Note that this property does not hold for the $\Box$ and $\Diamond$ modalities for the functor $FX = \mathcal{P}_f(A \times X)$. This can be seen via Figure 2, where the upward closure of $\{(b, 0)\}$ contains three elements. However, $\{(b, 0)\}$ is only contained in the modality $\Box_a$ (and no other modality), which does not coincide with the upward-closure of $\{(b, 0)\}$.

Next we show the following proposition, which immediately gives us a recipe for transforming cone modalities that satisfy the properties of Lemma 29 into the given modalities.

Proposition 30. *Given a set $\Lambda' \subseteq \Lambda$ of predicate liftings we have*

$$[\bigcap_{\lambda \in \Lambda'} \lambda]\varphi \equiv \bigwedge_{\lambda \in \Lambda'} [\lambda]\varphi.$$

An interesting question that we do not pursue further is whether it is possible to extract a minimal set of (monotone) predicate liftings from the set $F2$ with its order. How do we obtain "natural" modalities, such as $\Box$ and $\Diamond$? An additional question is how these extracted predicate liftings can be presented.

5 T-Beg: A Generic Tool for Games and the Construction of Distinguishing Formulas

5.1 Overview

A tool for playing bisimulation games is useful for teaching, for illustrating examples in talks, for case studies and in general for interaction with the user. There are already available tools, providing visual feedback to help the user to understand why two states are (not) bisimilar, such as The Bisimulation Game Game[5] or Bisimulation Games Tools[6] [8]. Both games are designed for labelled transition systems and [8] also covers branching bisimulation.

Our tool T-Beg goes beyond labelled transition system and allows to treat coalgebras in general (under the restrictions that we impose), that is, we exploit the categorical view to create a generic tool. As shown earlier in Sections 3 and 4, the coalgebraic game defined in Definition 2 provides us with a generic algorithm to compute the winning strategies and distinguishing formulas.

The user can either slip into the role of the spoiler or of the duplicator, playing on some coalgebra against the computer. The tool computes the winning strategy (if any) and follows this winning strategy if possible. We have also implemented the construction of the distinguishing formula for two non-bisimilar states.

The genericity over the functor is in practice achieved as follows: The user either selects an existing functor F, or implements his/her own functor by providing the code of one class with nine methods (explained below). Everything

[5] `http://www.brics.dk/bisim/`
[6] `https://www.jeroenkeiren.nl/blog/on-games-and-simulations/`

else, such as embedding the functor into the game and the visualization are automatically handled by T-Beg. In the case of weighted systems, T-Beg even handles the creation of the graphical representation.

Then, he/she enters or loads a coalgebra $\alpha : X \to FX$ (with X finite), stored as *csv* (comma separated value) file. Now the user can switch to the game view and start the game by choosing one of the two roles (spoiler or duplicator) and selecting a pair of states (x, y), based on the the visual graph representation.

Next, the computer takes over the remaining role and the game starts: In the game overview, the user is guided through the steps by using two colors to indicate whether it is spoiler's (violet) or duplicator's (cyan) turn (see Figure 6).

In the case of two non-bisimular states, the tool will display a distinguishing formula at the end of the game.

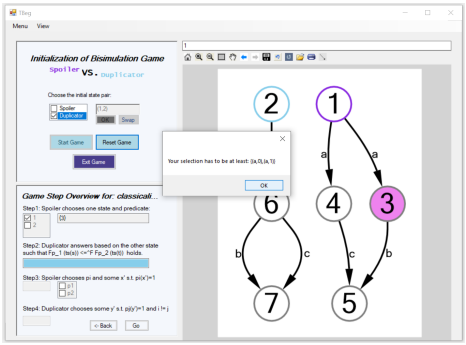

Fig. 6: Screenshot of the graphical user interface with a game being played.

5.2 Design

We now give an overview over the design and the relevant methods within the tool. We will also explain what has to be done in order to integrate a new functor.

T-Beg is a Windows tool offering a complete graphical interface, developed in Microsoft's Visual Studio using *C#*, especially *Generics*. The program is divided into the following five components: Model, View, Controller, Game and Functor. We have chosen *MVC (Model View Controller)* as a modular pattern, so modules can be exchanged. Here we have several *Model*$\langle T \rangle$ managed by the *Controller*, where the functor in the sense of a *Functor* class, which always implements the *Functor Interface*, is indicated by the parameter $\langle T \rangle$.

While the tool supports more general functors, there is specific support for functors F with $FX = V^{GX}$ where V specifies a semiring and GX is finite (whenever X finite). That is, F describes the branching type of a weighted transition system, where for instance $GX = A \times X + 1$ (introducing labels and termination). Coalgebras are of the form $X \to V^{GX}$ or – via currying – of the

form $X \times GX \to V$, which means that they can be represented by $X \times GX$-matrices (matrices with index sets X, GX). In the implementation V is the generic data type of the matrix entries. In the case of the powerset functor we simply have $V = 2$ and $GX = X$.

If the transition system can not simply be modelled as a matrix, there is an optional field that can be used to specify the system, since *Model*$\langle T \rangle$ calls the user-implemented method to initialize the transition system instance. The implementation of Algorithm 1 can be found in *Game*$\langle T, V \rangle$, representing the core of the tool's architecture.

Functor Interface As mentioned previously, the user has to provide nine methods in order to implement the functor: two are needed for the computation, two for rendering the coalgebra as a graph, one for creating modal logical formulas, another two for loading and saving, and two more for customizing the visual matrix representation.

We would like to emphasize here that the user is not expected to formally implement the functor in the sense of the categorical definition. In particular, we do not need the application of the functor to arrows, but we need methods that evaluate the conditions necessary for the game.

Within *MyFunctor*, which implements the interface *Functor*$\langle F, V \rangle$, the user defines the data structure F for the branching type of the transition system (e.g., a list or bit vector for powerset functor, or the corresponding function type in the case of the distribution functor). Further, the user specifies the type V that is needed to define the entries of $X \times GX$ (e.g. a double value for a weight or $0, 1$ to indicate the existence of a transition).

Then the following nine methods have to be provided:

Matrix$\langle F, V \rangle$*InitMatrix*$(\dots)$**:** This method is necessary to initialize the transition system with the string-based input of the user. The information about the states and the alphabet is provided via an input mask in the form of a matrix.

bool CheckDuplicatorsConditionStep2$(\dots)$: given two states x, y and two predicates p_1, p_2, this method checks whether

$$Fp_1(\alpha(x)) \leq^F Fp_2(\alpha(y)).$$

This method is used when playing the game (in Step 2) and in the partition refinement algorithm (Algorithm 1) for the case $p_1 = p_2$.

TSToGraph$(\dots)$: This method handles the implementation of the graph-based visualization of the transition system, via an external graph library[7]. In the case of weighted systems the user can trust the default implementation included within the *Model*. In this case, arrows between states and their labels are generated automatically.

GraphToTS$(\dots)$: This method is used for the other direction, i.e. to derive the transition system from a directed graph given by *Graph*.

[7] `https://www.nuget.org/packages/Microsoft.Msagl.GraphViewerGDI`

string GetModalityToString(...): This method is essential for the automatic generation of the modal logical formulas distinguishing two non-bisimilar states as described in Definition 17. In each call, the cone modality that results from $F\chi_P(\alpha(s))$ with $T(x, y) = (s, P)$ is converted into a string.
SaveTransitionSystem(...): In order to store a transition system in a *csv* file.
LoadTransitionSystem(...): In order to load a transition system from a *csv* file.
GetRowHeadings(...): T-Beg can visualize a transition system $\alpha : X \to FX$ as a $X \times GX$ matrix within a *DataGrid*. For this purpose, the user needs to specify how the *RowHeaders* are generated automatically.
ReturnRowCount(...): This method returns the number of rows of the matrix representing the coalgebra.

In addition, T-Beg uses a graph library[3], which in turn provides a *GraphEditor* that allows for storing graphs as *MSAGL* files or as *png* and *jpg* files.

6 Conclusion and Discussion

Our aim in this paper is to give concrete recipes for explaining non-bisimilarity in a coalgebraic setting. This involves the computation of the winning strategy of the spoiler in the bisimulation game, based on a partition refinement algorithm, as well as the generation of distinguishing formulas, following the ideas of [5]. Furthermore we have presented a tool that implements this functionality in a generic way. Related tools, as mentioned in [8], are limited to labelled transition systems and mainly focus on the spoiler strategy instead of generating distinguishing formulas.

In the future we would like to extend our prototype implementation to an efficient coalgebraic partition refinement algorithm, adapting the ideas of Kanellakis/Smolka [11] or Paige/Tarjan [15, 6]. The latter method achieves runtime $\mathcal{O}(n \cdot \log n)$, where n is the size of the system, by using so-called *three-way-splitting* that chooses equivalence classes for splitting in a clever way.

For the generation of distinguishing formulas an option would be to fix the modalities a priori and to use them in the game, similar to to the notion of λ-bisimulation [9, 14]. However, there might be infinitely many modalities and the partition refinement algorithm can not iterate over all of them. A possible solution would be to find a way to check the conditions symbolically in order to obtain suitable modalities.

Of course we are also interesting in whether can lift the extra asumptions that were necessary in order to re-code modalities in Section 4.3.

We are also interested in studying applications where we can exploit the fact that the distinguishing formula witnesses non-bisimilarity. For instance, we see interesting uses in the area of differential privacy [4], for which we would need to generalize the theory to a quantitative setting. That is, we would like to construct distinguishing formulas in the setting of quantitative coalgebraic logics, which characterizes behavioural distances.

Acknowledgements We thank Thorsten Wißmann for inspiring discussions on efficient coalgebraic partition refinement and zippability.

References

1. Balan, A., Kurz, A.: Finitary functors: From **Set** to **Preord** and **Poset**. In: Proc. of CALCO '11. pp. 85–99. Springer (2011), LNCS 6859
2. Baltag, A.: Truth-as-Simulation: Towards a Coalgebraic Perspective on Logic and Games. Tech. Rep. SEN-R9923, CWI (November 1999)
3. Barr, M.: Relational algebras. In: Proc. Midwest Category Seminar. LNM, vol. 137. Springer (1970)
4. Chatzikokolakis, K., Gebler, D., Palamidessi, C., Xu, L.: Generalized Bisimulation Metrics. In: Proc. of CONCUR '14. Springer (2014), LNCS/ARCoSS 8704
5. Cleaveland, R.: On Automatically Explaining Bisimulation Inequivalence. In: Proc. of CAV '90. pp. 364–372. Springer (1990), LNCS 531
6. Dorsch, U., Milius, S., Schröder, L., Wißmann, T.: Efficient Coalgebraic Partition Refinement. In: Proc. of CONCUR 2017. LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
7. Dorsch, U., Milius, S., Schröder, L., Wißmann, T.: Predicate Liftings and Functor Presentations in Coalgebraic Expression Languages. CoRR **abs/1805.07211** (2018)
8. de Frutos-Escrig, D., Keiren, J.J.A., Willemse, T.A.C.: Games for Bisimulations and Abstraction. CoRR **abs/1611.00401** (2016)
9. Gorín, D., Schröder, L.: Simulations and Bisimulations for Coalgebraic Modal Logics. In: Proc. of CALCO '13. pp. 253–266. Springer (2013), LNCS 8089
10. Hennessy, M., Milner, A.: Algebraic Laws for Nondeterminism and Concurrency. J. ACM **32**(1), 137–161 (Jan 1985)
11. Kanellakis, P.C., Smolka, S.A.: CCS Expressions, Finite State Processes, and Three Problems of Equivalence. Inf. Comput. **86**, 43–68 (1990)
12. Komorida, Y., Katsumata, S.Y., Hu, N., Klin, B., Hasuo, I.: Codensity Games for Bisimilarity. In: Proc. of LICS '19. pp. 1–13. ACM (2019)
13. König, B., Küpper, S.: A Generalized Partition Refinement Algorithm, Instantiated to Language Equivalence Checking for Weighted Automata. Soft Computing **22**(4) (2018)
14. König, B., Mika-Michalski, C.: (Metric) Bisimulation Games and Real-Valued Modal Logics for Coalgebras. In: Proc. of CONCUR '18. LIPIcs, vol. 118, pp. 37:1–37:17. Schloss Dagstuhl – Leibniz Center for Informatics (2018)
15. Paige, R., Tarjan, R.E.: Three Partition Refinement Algorithms. SIAM J. Comput. **16**(6), 973–989 (1987)
16. Pattinson, D.: Coalgebraic modal logic: soundness, completeness and decidability of local consequence. Theoretical Computer Science **309**(1), 177 – 193 (2003)
17. Rutten, J.: Universal coalgebra: a theory of systems. Theoretical Computer Science **249**(1), 3 – 80 (2000)
18. Schröder, L.: Expressivity of Coalgebraic Modal Logic: The Limits and Beyond. Theoretical Computer Science **390**(2), 230 – 247 (2008)
19. Stirling, C.: Bisimulation, modal logic and model checking games. Logic Journal of the IGPL **7**(1), 103–124 (01 1999)
20. Trnková, V.: General theory of relational automata. Fund. Inform. **3**, 189–234 (1980)

A Proofs

Proposition 7. *Assume that $R_n = \nu\mathcal{F}_\alpha, T, I$ have been computed by Algorithm 1. Furthermore let $(x, y) \notin R_n$, which means that $I(x, y) < \infty$ and $T(x, y)$ is defined. Then the following constitutes a winning strategy for the spoiler:*

- *Let $T(x, y) = (s, P_1)$. Then in Step 1 S plays a predicate $p_1 = \chi_{P_1}$ and $s \in \{x, y\}$.*
- *Assume that in Step 2 D answers with a state t and a predicate p_2 such that $Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t))$.*
- *Then, in Step 3 there exists a state $y' \in X$ such that $p_2(y') = 1$ and $I(x', y') < I(x, y)$ for all $x' \in X$ with $p_1(x') = 1$. S will hence select p_2 and this state y'.*
- *Next, in Step 4 D selects some x' with $p_1(x') = 1$ and the game continues with $(x', y') \notin R_n$.*

Proof. We have to show that whenever we reach Step 3 there always exists a state $y' \in X$ such that $p_2(y') = 1$ and $I(x', y') < I(x, y)$ for all $x' \in X$ with $p_1(x') = 1$.

Let us first observe that $p_2 \not\leq p_1$. If this were the case, we would have $Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t)) \leq^F Fp_1(\alpha(t))$. But $\{x, y\} = \{s, t\}$ are separated at Step $I(x, y) = i$ precisely because this inequality does not hold for p_1 which represents one of the equivalence classes of R_{i-1}.

Hence there exists an $y' \in X$ such that $p_2(y') = 1$ and $p_1(y') = 0$.

Since the equivalence relations R_i are subsequently refined by the algorithm, p_1 – being an equivalence class of R_{i-1} – is a union of equivalence classes of R_n. So, since y' is not contained in $P_1 = \hat{p_1}$, it is not in R_{i-1}-relation to any $x' \in P_1$, hence $I(x', y') \leq i - 1$ for all such x'.

Since the index $I(x, y)$ decreases after every round of the game, D will eventually not be able to find a suitable answer in Step 2 and will lose. □

Theorem 10. *Let $\alpha : X \to FX$ be a coalgebra where F is separable by singletons. It holds that $\nu\mathcal{F}_\alpha = W_\alpha$, i.e., it contains exactly the pairs $(x, y) \in X \times X$ for which the duplicator has a winning strategy.*

Proof.

"⊆" Assume that $(x, y) \in \nu\mathcal{F}_\alpha = R_n$. We show that $x \sim y$ and with [14] it follows that $(x, y) \in W_\alpha$. We do this by constructing a coalgebra homomorphism f with $f(x) = f(y)$.
Let $Y = E(R_n)$, the set of equivalence classes of R_n and we define $f\colon X \to Y$, $f(x) = [x]_{R_n}$. In order to show that f is a coalgebra homomorphism, we have to construct a coalgebra $\beta\colon Y \to FY$ such that $\beta \circ f = Ff \circ \alpha$.
We define $\beta([x]_{R_n}) = Ff(\alpha(x))$ and it suffices to show that β is well-defined. So let $(x, y) \in R_n$ and assume by contradiction that $t_1 = Ff(\alpha(x)) \neq Ff(\alpha(y)) = t_2$. Then, since F is separable by singletons, we have a singleton predicate p with $Fp(t_1) \neq Fp(t_2)$. By expanding the definition we get $F(p \circ$

$f)(\alpha(x)) \neq F(p \circ f)(\alpha(y))$. By construction $p \circ f = \chi_P$, where P is an equivalence class of R_n. This is a contradiction, since $\mathcal{F}_\alpha(R_{n-1}) = R_n = R_{n-1}$, which indicates that there can not be such a P.

"$\supseteq$" Whenever $(x, y) \notin \nu\mathcal{F}_\alpha = R_n$, we have shown in Proposition 7 that the spoiler has a winning strategy, which implies $(x, y) \notin W_\alpha$. Hence $W_\alpha \subseteq \nu\mathcal{F}_\alpha$. □

Lemma 13. *Assume that $F\colon \boldsymbol{Set} \to \boldsymbol{Set}$ is a functor preserving injections. We have that if F is separable by singletons, it is m-zippable. Furthermore if F is m-zippable, then it is separable by singletons for all sets X with $|X| \leq m$.*

Proof.

– Suppose that F is separable by singletons. We need to show that

$$F(A_1 + \cdots + A_m) \xrightarrow{\langle F(f_1),\ldots,F(f_m)\rangle} F(A_1 + 1) \times \cdots \times F(A_m + 1)$$

is injective. Hence let $t_1, t_2 \in F(A_1 + \cdots + A_m)$ with

$$\langle F(f_1), \ldots, F(f_m)\rangle(t_1) = \langle F(f_1), \ldots, F(f_m)\rangle(t_2)$$

be given. The situation is depited in Figure 7 below.
Now let $x_i \in A_i$ and consider the singleton predicate $\chi_{\{x_i\}}\colon A_1 + \cdots + A_m \to 2$, which decomposes as $\chi_{\{x_i\}} = h_{x_i} \circ f_i$ where $h_{x_i}\colon A_i + 1 \to 2$ is the characteristic function of x_i on $A_i + 1$ (see Figure 8 below).

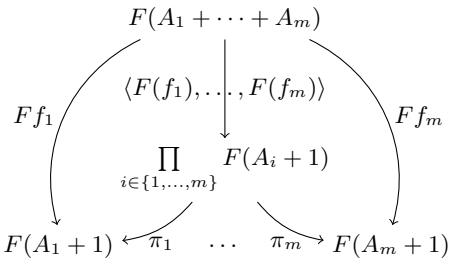

Fig. 7

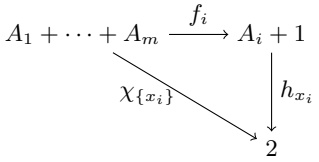

Fig. 8

Now we can proceed as follows:

$$\begin{aligned}
& \pi_i(\langle F(f_1), \ldots, F(f_m)\rangle(t_1)) = \pi_i(\langle F(f_1), \ldots, F(f_m)\rangle(t_2)) \\
\Rightarrow \quad & F(f_i)(t_1) = F(f_i)(t_2) \\
\Rightarrow \quad & Fh_{x_i}(F(f_i)(t_1)) = Fh_{x_i}(F(f_i)(t_2)) \\
\Rightarrow \quad & F\chi_{\{x_i\}}(t_1) = F\chi_{\{x_i\}}(t_2)
\end{aligned}$$

Since this holds for all x_i in $A_1 + \cdots + A_m$, and F is separable by singletons, we can conclude that $t_1 = t_2$.

– We first observe that every functor that is m-zippable is also m'-zippable for $m' \leq m$ (just set $A_i = \emptyset$ for some i). Hence it is sufficient to prove that whenever F is m-zippable, then it is separable by singletons for all sets X with $|X| = m$. So we can assume without loss of generality that $X = \{x_1, \ldots, x_m\}$.
We set $A_i = \{x_i\}$ and know from the premise that

$$F(A_1 + \cdots + A_m) \xrightarrow{\langle F(f_1),\ldots,F(f_m)\rangle} F(A_1 + 1) \times \cdots \times F(A_m + 1)$$

is injective (see Figure 7).
Let $t_1, t_2 \in FX$ and $t_1 \neq t_2$ be given. Due to the injectivity of the map above, we know that there exists an index i such that $Ff_i(t_1) \neq Ff_i(t_2)$. Since $A_i + 1 \cong 2$, every f_i is itself a singleton predicate and hence we witness the inequality of t_1, t_2 via a singleton. □

Proposition 18. *Let $\alpha : X \to FX$ be a coalgebra and assume that we have computed R_n, T, I with Algorithm 1. Then, given $(x, y) \notin R_n$, the construction in Definition 17 yields a formula $\varphi_{x,y} \in \mathcal{L}^\kappa(\Lambda)$ such that $x \vDash \varphi_{x,y}$ and $y \nvDash \varphi_{x,y}$.*

Proof. We prove this by induction over $i = I(x, y)$:

$i = 1$: x, y have been separated at Step 1, since $F\chi_X(\alpha(x)) \not\leq^F F\chi_X(\alpha(y))$, where $T(x, y) = (x, X)$ (or vice versa), because X is the only equivalence class so far. Note also that $[\![tt]\!] = X$.
We set $v = F\chi_X(\alpha(x))$, $\lambda = \uparrow v$ and we have

$$[\![\varphi_{x,y}]\!](x) = \lambda(F[\![\varphi]\!](\alpha(x))) = \lambda(F\chi_X(\alpha(x))) = \lambda(t) = 1$$
$$[\![\varphi_{x,y}]\!](y) = \lambda(F[\![\varphi]\!](\alpha(y))) = \lambda(F\chi_X(\alpha(y))) = 0$$

Hence $x \models \varphi_{x,y}$ and $y \not\models \varphi_{x,y}$.
In the case where $T(x, y) = (y, X)$, we have $v = F\chi_X(\alpha(y))$, $\lambda = \uparrow v$ and we obtain

$$[\![[\lambda]\varphi]\!](x) = \lambda(F[\![\varphi]\!](\alpha(x))) = \lambda(F\chi_X(\alpha(x))) = 0$$
$$[\![[\lambda]\varphi]\!](y) = \lambda(F[\![\varphi]\!](\alpha(y))) = \lambda(F\chi_X(\alpha(y))) = \lambda(v) = 1$$

Hence again $x \models \varphi_{x,y}$ and $y \not\models \varphi_{x,y}$.

$i \to i + 1$: Due to the induction hypothesis we can assume that the $\varphi_{x',y'}$ are distinguishing formulas for (x', y') with $I(x', y') < i + 1$.
First, we show that $[\![\varphi]\!] = P$.
  – Let $z \in P$. Then there exists an $x' \in P$ (namely $x' = z$) such that $z \models \varphi_{x',y'}$ for all $y' \notin P$. Furthermore, by construction of $\varphi_{x',y'}$ it holds that $y' \not\models \varphi_{x',y'}$. This means that $x \models \bigwedge_{y' \in X \setminus P} \varphi_{x',y'}$ and also $x \models \bigvee_{x' \in P} \bigwedge_{y' \in X \setminus P} \varphi_{x',y'} = \varphi$.
  – Let $z \notin P$. Then for every $x' \in P$ there exists an $y' \notin P$ (namely $y' = z$) such that $z \not\models \varphi_{x',y'}$. Hence $z \not\models \bigwedge_{y' \in X \setminus P} \varphi_{x',y'}$. Since this is true for every such x' we also have $x \not\models \bigvee_{x' \in P} \bigwedge_{y' \in X \setminus P} \varphi_{x',y'} = \varphi$.

Assume that $T(x,y) = (x,P)$ (the case $T(x,y) = (y,P)$ can be handled analogously as for $i = 1$). Hence we know that $F\chi_P(\alpha(x)) \not\leq^F F\chi_P(\alpha(y))$. We set $v = F\chi_P(\alpha(x))$, $\lambda = \uparrow v$ and we have

$$[\![\varphi_{x,y}]\!](x) = \lambda(F[\![\varphi]\!](\alpha(x))) = \lambda(F\chi_P(\alpha(x))) = \lambda(v) = 1$$
$$[\![\varphi_{x,y}]\!](y) = \lambda(F[\![\varphi]\!](\alpha(y))) = \lambda(F\chi_P(\alpha(y))) = 0$$

Hence $x \models \varphi_{x,y}$ and $y \not\models \varphi_{x,y}$.

□

Lemma 19. *Given two states $(x,y) \notin R_n$ and a distinguishing formula $\varphi_{x,y}$ based on Definition 17. Let (x',y') be given such that $I(x',y') > I(x,y)$. Then $x' \vDash \varphi_{x,y}$ if and only if $y' \vDash \varphi_{x,y}$.*

Proof. We have to distinguish two different cases for $I(x',y')$

$I(x',y') = 1$: this can not be true since we require $I(x',y') > I(x,y) \geq 1$.

$I(x',y') > 1$: For any (x,y) with $I(x,y) < I(x',y')$ we have $x \vDash \varphi_{x,y}$ and $y \nvDash \varphi_{x,y}$ where $\varphi_{x,y} = [\lambda]\varphi$, $\lambda = \uparrow F\chi_P(\alpha(x))$ and $T(x,y) = (x,P)$ (the case $T(x,y) = (y,P)$ is analogous). Furthermore, the semantics of φ is $[\![\varphi]\!] = \chi_P$ (for details we refer to the proof of Proposition 18). Now, assume without loss of generality that the following holds

$$1 = [\![\varphi_{x,y}]\!](x') = \lambda(F\chi_P(\alpha(x')))$$
$$0 = [\![\varphi_{x,y}]\!](y') = \lambda(F\chi_P(\alpha(y')))$$

Due to Proposition 4 λ is monotone. Therefore, the above assumption implies $F\chi_P(\alpha(x')) \not\leq^F F\chi_P(\alpha(y'))$. But this yields a contradiction, since then x', y' would have been separated in a Step $i \leq I(x,y) < I(x',y')$. □

Lemma 20. *Let $(x,y) \notin R_i$ and let P be an equivalence class of R_{i-1}. Furthermore let*

$$\varphi' = \bigwedge_{y' \in\ X \setminus P} \varphi_{x',y'}$$

for some $x' \in P$. Then $[\![\varphi']\!] = \chi_P$.

Proof. Clearly $[\![\varphi']\!] \leq [\![\varphi]\!] = \chi_P$.

We now have to show that the other inequality holds as well, so let $z \in P$. Furthermore let y' be arbitrary such that $y' \notin P$. Since $z, x' \in P$ and $y' \notin P$, where P is an equivalence class, we know that $I(z,x') > I(x',y')$ (possibly even $I(z,x') = \infty$). Hence, by Lemma 19 we have that $z \models \varphi_{x',y'}$ if and only if $x' \models \varphi_{x',y'}$. And since the latter holds, we have $z \models \varphi_{x',y'}$.

Hence $z \models \bigwedge_{y' \in X \setminus P} \varphi_{x',y'} = \varphi$. In summary, we get $\chi_P \leq [\![\varphi]\!]$.

□

Lemma 25. *Let Λ be set of predicate liftings. Furthermore we denote the four functions on 2 by id_2, one (constant 1-function), zero (constant 0-function) and neg ($neg(0) = 1$, $neg(1) = 0$).*

Then

$$\Lambda' = \{\lambda, \lambda \circ F one, \lambda \circ F zero, \lambda \circ F neg \mid \lambda \in \Lambda\}$$

is a set of strongly separating predicate liftings.

Furthermore for every formula φ we have that

$$[\lambda \circ F one]\varphi \equiv [\lambda]tt \qquad [\lambda \circ F zero]\varphi \equiv [\lambda]ff \qquad [\lambda \circ F neg]\varphi \equiv [\lambda](\neg\varphi)$$

Proof. Let $t_1, t_2 \in F2$ with $t1 \neq t_2$. According to the definition of strong separation there must be a predicate $p\colon 2 \to 2$ such that $\lambda(Fp(t_1)) \neq \lambda(Fp(t_2))$. Since there are only four such functions, p must be one of id_2, *one*, *zero*, *neg* and we immediately obtain that Λ' is strongly separating.

In addition we have that, given a coalgebra $\alpha\colon X \to FX$:

$$\begin{aligned}
[\![[\lambda \circ F one]\varphi]\!] &= \lambda \circ F one \circ F[\![\varphi]\!] \circ \alpha = \lambda \circ F(one \circ [\![\varphi]\!]) \circ \alpha \\
&= \lambda \circ F[\![tt]\!] \circ \alpha = [\![[\lambda]tt]\!] \\
[\![[\lambda \circ F zero]\varphi]\!] &= \lambda \circ F zero \circ F[\![\varphi]\!] \circ \alpha = \lambda \circ F(zero \circ [\![\varphi]\!]) \circ \alpha \\
&= \lambda \circ F[\![ff]\!] \circ \alpha = [\![[\lambda]ff]\!] \\
[\![[\lambda \circ F neg]\varphi]\!] &= \lambda \circ F neg \circ F[\![\varphi]\!] \circ \alpha = \lambda \circ F(neg \circ [\![\varphi]\!]) \circ \alpha \\
&= \lambda \circ F[\![\neg\varphi]\!] \circ \alpha = [\![[\lambda]\neg\varphi]\!]
\end{aligned}$$

□

Proposition 26. *Assume that $F2$ is finite and let Λ be a strongly separating set of predicate liftings. Let furthermore $v \in F2$ and let φ be a formula.*

For a given $u \in F2$ we write $\Lambda_u = \{\lambda \in \Lambda \mid \lambda(u) = 1\}$. Then it holds that

$$[\uparrow v]\varphi \equiv \bigvee_{v \leq^F u} \Big(\bigwedge_{\lambda \in \Lambda_u} [\lambda]\varphi \wedge \bigwedge_{\lambda \notin \Lambda_u} \neg[\lambda]\varphi \Big)$$

Proof. First observe that since Λ is strongly separating, every $u \in F2$ is characterized uniquely by Λ_u.

Let $\alpha\colon X \to FX$ be a coalgebra. We set $\psi_u = \bigwedge_{\lambda \in \Lambda_u} [\lambda]\varphi \wedge \bigwedge_{\lambda \notin \Lambda_u} \neg[\lambda]\varphi$ and we first show that

$$x \models \psi_u \iff u = F[\![\varphi]\!](\alpha(x))$$

⇒: Assume that $x \models \psi_u$. This means that for every $\lambda \in \Lambda_u$ we have that $\lambda(F[\![\varphi]\!](\alpha(x))) = 1$ and for every $\lambda \notin \Lambda_u$ we have that $\lambda(F[\![\varphi]\!](\alpha(x))) = 0$. This means that u and $F[\![\varphi]\!](\alpha(x))$ are both characterized by Λ_u and from the strong separation property it follows that they are equal, i.e., $u = F[\![\varphi]\!](\alpha(x))$.

⇐: Assume that $u = F[\![\varphi]\!](\alpha(x))$. Then for every $\lambda \in \Lambda_u$ we have that $[\![[\lambda]\varphi]\!](x) = \lambda(F[\![\varphi]\!](\alpha(x))) = \lambda(u) = 1$. For every $\lambda \notin \Lambda_u$ we obtain $[\![[\lambda]\varphi]\!](x) = 0$. Everything combined, we have $[\![\psi_u]\!](x) = 1$ and hence $x \models \psi_u$.

We can conclude the proof by observing that

$$\begin{aligned}
&x \models [\uparrow v]\varphi \iff v \leq^F F[\![\varphi]\!](\alpha(x)) \iff \exists u\colon \big(v \leq^F u \wedge u = F[\![\varphi]\!](\alpha(x))\big) \\
&\iff \exists u\colon \big(v \leq^F u \wedge x \models \psi_u\big) \iff x \models \bigvee_{v \leq^F u} \psi_u
\end{aligned}$$

□

Lemma 29. *Let F be the functor with $FX = (\mathcal{D}X + 1)^A$ and let Λ be the separating set of predicate liftings from Example 28. Furthermore let $v \in F2$. Then it holds that:*

$$\uparrow v = \bigcap_{\lambda \in \Lambda, \lambda(v)=1} \lambda$$

Proof.

"$\subseteq$" Let $u \in F2$ with $t \in \uparrow u$, i.e., $v \leq^F u$. Whenever $\lambda(v) = 1$ we also have $\lambda(u) = 1$ due to the monotonicity of the predicate liftings (cf. Proposition 3) and hence $u \in \hat{\lambda}$. Since this holds for all such λ, we can conclude that $u \in \bigcap_{\lambda \in \Lambda, \lambda(v)=1} \hat{\lambda}$.

"$\supseteq$" Now suppose by contradiction that we have $u \in F2$ with $v \not\leq^F u$ and $u \in \bigcap_{\lambda \in \Lambda, \lambda(v)=1} \hat{\lambda}$.

There are three cases which may cause $v \not\leq^F u$, in particular they are distinguished by a specific $a \in A$:

- $v(a), u(a) \in \mathbb{R}$, but $v(a) \not\leq u(a)$, which implies $u(a) < v(a)$. However, there exists $q \in [0,1] \cap \mathbb{Q}$ with $u(a) < q \leq v(a)$ and for the corresponding modality $\lambda_{(a,q)} \in \Lambda$ we have $\lambda_{(a,q)}(u) = 0$, $\lambda_{(a,q)}(v) = 1$ and hence $u \notin \bigcap_{\lambda \in \Lambda, \lambda(v)=1} \hat{\lambda}$.
- $v(a) \in \mathbb{R}$, $u(a) = \bullet$: Now take any $q \in [0,1] \cap \mathbb{Q}$ with $q \leq v(a)$. We use the modality $\lambda_{(a,q)}$, for which we have $\lambda_{(a,q)}(u) = 0$, $\lambda_{(a,q)}(v) = 1$ and the proof proceeds as before.
- $v(a) = \bullet$, $u(a) \in \mathbb{R}$: Now we take the modality $\lambda_{(a,\bullet)}$, for which we have $\lambda_{(a,\bullet)}(u) = 0$, $\lambda_{(a,\bullet)}(v) = 1$ and again the proof proceeds as before.

□

Proposition 30. *Given a set $\Lambda' \subseteq \Lambda$ of predicate liftings we have*

$$[\bigcap_{\lambda \in \Lambda'} \lambda]\varphi \equiv \bigwedge_{\lambda \in \Lambda'} [\lambda]\varphi.$$

Proof.

"$\subseteq$" Let $x \vDash [\bigcap_{\lambda \in \Lambda'} \lambda]\varphi$, which implies that $(\bigcap_{\lambda \in \Lambda'} \lambda)(F[\![\varphi]\!](\alpha(x))) = 1$. From this we conclude that $\lambda(F[\![\varphi]\!](\alpha(x))) = 1$ for all $\lambda \in \Lambda'$, $x \models [\lambda]\varphi$. And finally we have $x \models \bigwedge_{\lambda \in \Lambda'} [\lambda]\varphi$.

"$\supseteq$" Let $x \vDash \bigwedge_{\lambda \in \Lambda'} [\lambda]\varphi$, which means that $x \vDash [\lambda]\varphi$ for all $\lambda \in \Lambda'$. This implies that $\lambda(F[\![\varphi]\!](\alpha(x))) = 1$. Hence we obtain $(\bigcap_{\lambda \in \Lambda'} \lambda)(F[\![\varphi]\!](\alpha(x))) = 1$ and finally $x \vDash [\bigcap_{\lambda \in \Lambda'} \lambda]\varphi$. □