Explaining Behavioural Inequivalence Generically in Quasilinear Time



Thorsten Wißmann Stefan Milius Lutz Schröder

Radboud University, Nijmegen

Friedrich-Alexander-Universität Erlangen-Nürnberg

Concur 2021

Explaining
Behavioural
Inequivalence

Generically

In
Quasilinear
Time

Explaining
Behavioural
Inequivalence

Generically

In
Quasilinear
Time

Explaining Behavioural Inequivalence

Generically

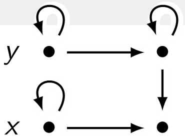
In Quasilinear Time

Explaining Behavioural Inequivalence

Generically

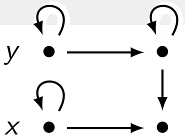
In Quasilinear Time

Why are x and y not bisimilar?



Explaining Behavioural Inequivalence

Why are x and y not bisimilar?



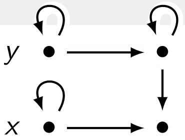
$\Rightarrow$ because $\Box\Diamond\top$ distinguishes them

Generically

In Quasilinear Time

Explaining Behavioural Inequivalence

Why are x and y not bisimilar?



$\Rightarrow$ because $\Box\Diamond\top$ distinguishes them

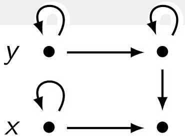
$\Rightarrow$ algorithmic construction?

Generically

In Quasilinear Time

Explaining Behavioural Inequivalence

Why are x and y not bisimilar?



$\Rightarrow$ because $\Box\Diamond\top$ distinguishes them

$\Rightarrow$ algorithmic construction?

Generically

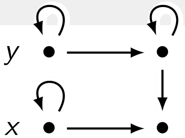
For different system types

- weighted or probabilistic systems
- (tree) automata

In Quasilinear Time

Explaining Behavioural Inequivalence

Why are x and y not bisimilar?



$\Rightarrow$ because $\Box\Diamond\top$ distinguishes them

$\Rightarrow$ algorithmic construction?

Generically

For different system types

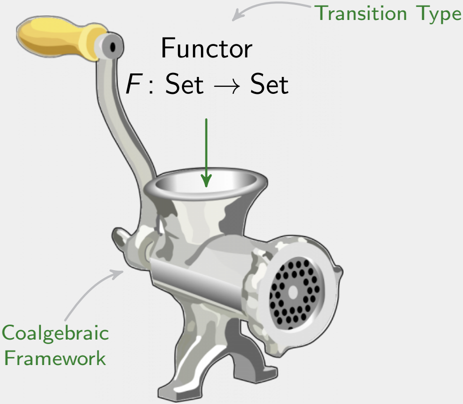
- weighted or probabilistic systems
- (tree) automata

In Quasilinear Time

$$\mathcal{O}(\overset{\text{transitions}}{m} \cdot \log \overset{\text{states}}{n}) \begin{cases} \text{Formula dag size} \\ \text{Run time} \end{cases}$$

Coalgebra – Generic state based systems

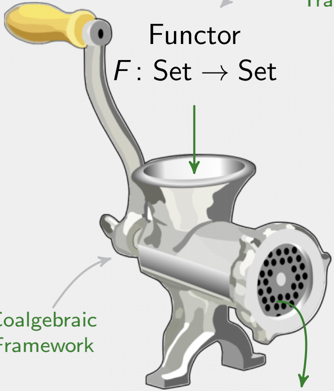
Coalgebra – Generic state based systems



Coalgebra – Generic state based systems

Transition Type

Functor
$F\colon \mathsf{Set} \to \mathsf{Set}$



Coalgebraic Framework

F-Coalgebra
$C \xrightarrow{c} FC$

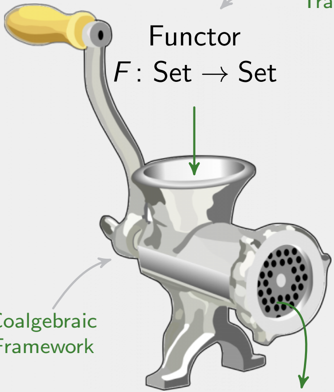
Coalgebra – Generic state based systems

Powerset $\mathcal{P}$

Transition Type

Functor

$F\colon \mathsf{Set} \to \mathsf{Set}$



Coalgebraic Framework

F-Coalgebra

$C \xrightarrow{c} FC$

Transition System

$C \xrightarrow{c} \mathcal{P}C$

Coalgebra – Generic state based systems

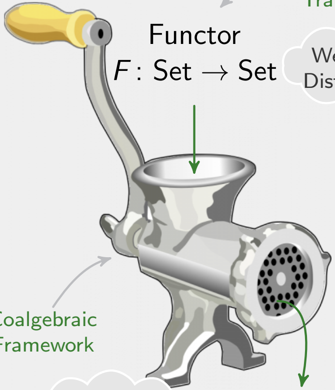
Powerset $\mathcal{P}$

Transition Type

Functor

$F\colon \mathsf{Set} \to \mathsf{Set}$

Weights $\mathbb{R}^{(-)}$
Distributions $\mathcal{D}$



Coalgebraic Framework

Markov Chain
$C \xrightarrow{c} \mathbb{R}^C$

Transition System
$C \xrightarrow{c} \mathcal{P}C$

F-Coalgebra

$C \xrightarrow{c} FC$

Coalgebra – Generic state based systems

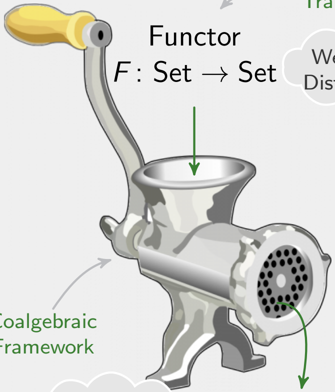
Powerset $\mathcal{P}$

Transition Type

Automata $2 \times (-)^A$

Functor
$F\colon \mathsf{Set} \to \mathsf{Set}$

Weights $\mathbb{R}^{(-)}$
Distributions $\mathcal{D}$



Coalgebraic Framework

Markov Chain $C \xrightarrow{c} \mathbb{R}^C$

Transition System $C \xrightarrow{c} \mathcal{P}C$

F-Coalgebra
$C \xrightarrow{c} FC$

Coalgebra – Generic state based systems

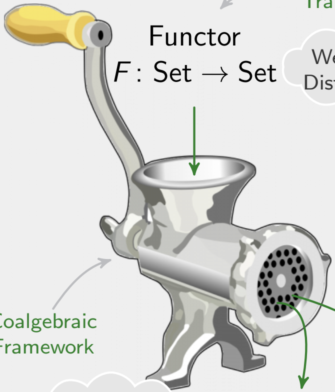
Powerset $\mathcal{P}$

Transition Type

Functor
$F\colon \mathsf{Set} \to \mathsf{Set}$

Weights $\mathbb{R}^{(-)}$
Distributions $\mathcal{D}$

Automata
$2 \times (-)^A$



Coalgebraic Framework

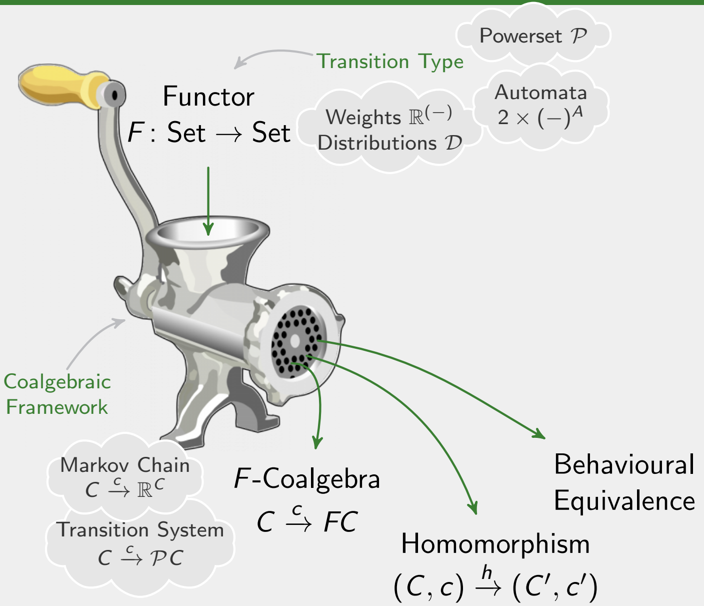
Markov Chain
$C \xrightarrow{c} \mathbb{R}^C$

Transition System
$C \xrightarrow{c} \mathcal{P}C$

F-Coalgebra
$C \xrightarrow{c} FC$

Homomorphism
$(C, c) \xrightarrow{h} (C', c')$

Coalgebra – Generic state based systems



Coalgebra – Generic state based systems

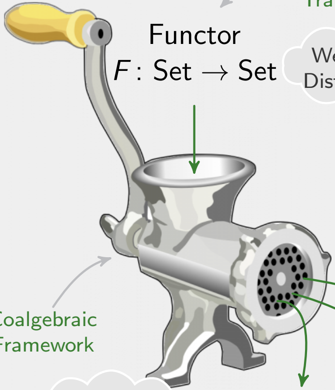
Powerset $\mathcal{P}$

Transition Type

Functor
$F\colon \mathsf{Set} \to \mathsf{Set}$

Weights $\mathbb{R}^{(-)}$
Distributions $\mathcal{D}$

Automata
$2 \times (-)^A$



Coalgebraic Framework

Markov Chain
$C \xrightarrow{c} \mathbb{R}^C$

Transition System
$C \xrightarrow{c} \mathcal{P}C$

F-Coalgebra
$C \xrightarrow{c} FC$

Homomorphism
$(C, c) \xrightarrow{h} (C', c')$

Behavioural Equivalence

Strong Bisimilarity

Language Equivalence

Coalgebra – Generic state based systems

Powerset $\mathcal{P}$

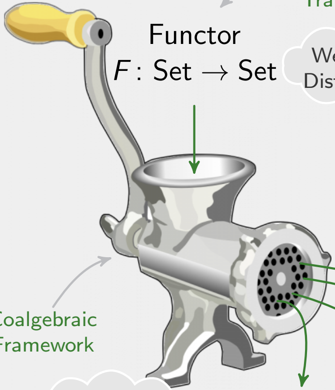
Transition Type

Functor

$F\colon \mathsf{Set} \to \mathsf{Set}$

Weights $\mathbb{R}^{(-)}$
Distributions $\mathcal{D}$

Automata
$2 \times (-)^A$



Coalgebraic Framework

Coalgebraic Logic

Behavioural Equivalence

Strong Bisimilarity

Language Equivalence

Markov Chain
$C \xrightarrow{c} \mathbb{R}^C$

Transition System
$C \xrightarrow{c} \mathcal{P}C$

F-Coalgebra

$C \xrightarrow{c} FC$

Homomorphism

$(C, c) \xrightarrow{h} (C', c')$

Coalgebraic Logic

Syntax

e.g. $\Lambda = \{\Diamond/1\}$ for $FX = \mathcal{P}X$

For a modal signature Λ, put

$$\phi_1, \ldots, \phi_n ::= \top \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \heartsuit(\phi_1, \ldots, \phi_n) \qquad \heartsuit/n \in \Lambda$$

Semantics in $c\colon C \to FC$

$$[\![\top]\!] = C \qquad [\![\phi \wedge \psi]\!] = [\![\phi]\!] \cap [\![\psi]\!] \qquad [\![\neg\phi]\!] = C \setminus [\![\phi]\!]$$

Coalgebraic Logic

Syntax

e.g. $\Lambda = \{\Diamond/1\}$ for $FX = \mathcal{P}X$

For a modal signature Λ, put

$$\phi_1, \ldots, \phi_n ::= \top \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \heartsuit(\phi_1, \ldots, \phi_n) \qquad \heartsuit/n \in \Lambda$$

Definition: Predicate Lifting for $\heartsuit/n \in \Lambda$

Pattinson '03

$=$ Natural transformation $[\![\heartsuit]\!]_X \colon (2^X)^n \to 2^{FX}$

Semantics in $c\colon C \to FC$

$$[\![\top]\!] = C \qquad [\![\phi \wedge \psi]\!] = [\![\phi]\!] \cap [\![\psi]\!] \qquad [\![\neg\phi]\!] = C \setminus [\![\phi]\!]$$

Coalgebraic Logic

Syntax e.g. $\Lambda = \{\Diamond/1\}$ for $FX = \mathcal{P}X$

For a modal signature Λ, put

$$\phi_1, \dots, \phi_n ::= \top \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \heartsuit(\phi_1, \dots, \phi_n) \qquad \heartsuit/n \in \Lambda$$

Definition: Predicate Lifting for $\heartsuit/n \in \Lambda$ Pattinson '03

$=$ Natural transformation $[\![\heartsuit]\!]_X \colon (2^X)^n \to 2^{FX}$

e.g. $[\![\Diamond]\!]_X \colon 2^X \to 2^{\mathcal{P}X} \qquad P \mapsto \{K \in \mathcal{P}X \mid \exists x \in K \colon x \in P\}$

Semantics in $c\colon C \to FC$

$$[\![\top]\!] = C \qquad [\![\phi \wedge \psi]\!] = [\![\phi]\!] \cap [\![\psi]\!] \qquad [\![\neg\phi]\!] = C \setminus [\![\phi]\!]$$

Coalgebraic Logic

Syntax e.g. $\Lambda = \{\Diamond/1\}$ for $FX = \mathcal{P}X$

For a modal signature Λ, put

$$\phi_1, \dots, \phi_n ::= \top \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \heartsuit(\phi_1, \dots, \phi_n) \qquad \heartsuit/n \in \Lambda$$

Definition: Predicate Lifting for $\heartsuit/n \in \Lambda$ Pattinson '03

$=$ Natural transformation $[\![\heartsuit]\!]_X \colon (2^X)^n \to 2^{FX}$

e.g. $[\![\Diamond]\!]_X \colon 2^X \to 2^{\mathcal{P}X} \qquad P \mapsto \{K \in \mathcal{P}X \mid \exists x \in K \colon x \in P\}$

Semantics in $c \colon C \to FC$

$$[\![\top]\!] = C \qquad [\![\phi \wedge \psi]\!] = [\![\phi]\!] \cap [\![\psi]\!] \qquad [\![\neg\phi]\!] = C \setminus [\![\phi]\!]$$

$$x \in [\![\heartsuit(\phi_1, \dots, \phi_n)]\!] \quad :\Leftrightarrow \quad c(x) \in [\![\heartsuit]\!]_C([\![\phi_1]\!], \dots, [\![\phi_n]\!])$$

Witnesses for (In-)Equivalence

Task
Given x, y in a finite transition system, construct a witness whether x, y are bisimilar

Yes → find bisimulation relating x and y

No → find modal formula distinguishing x and y

Adequacy Theorem for transition systems

x, y bisimilar $\Longrightarrow$ x, y accept the same modal formulae

Witnesses for (In-)Equivalence

Task

Given x, y in a finite transition system, construct a witness whether x, y are bisimilar

Yes: find bisimulation relating x and y

No: find modal formula distinguishing x and y

Adequacy Theorem for F-coalgebras

x, y beh. equivalent $\Longrightarrow$ x, y accept the same coalgebraic formulae

Witnesses for (In-)Equivalence

Task

Given x, y in a finite F-coalgebra, construct a witness whether x, y are behavourally equivalent

Yes

find coalgebra morphism relating x and y

No

find coalgebraic formula distinguishing x and y

Adequacy Theorem for F-coalgebras

x, y beh. equivalent $\implies$ x, y accept the same coalgebraic formulae

Witnesses for (In-)Equivalence

Task

Given x, y in a finite F-coalgebra, construct a witness whether x, y are behavourally equivalent

Yes → find coalgebra morphism relating x and y

coalgebraic partition refinement (Concur'17)

No → find coalgebraic formula distinguishing x and y

the present paper

Adequacy Theorem for F-coalgebras

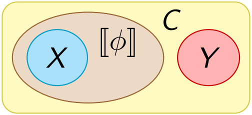
x, y beh. equivalent $\implies$ x, y accept the same coalgebraic formulae

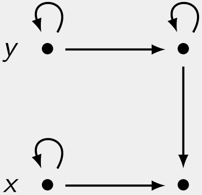
Certificates

For $c\colon C \to FC$, e.g. a transition system with states C

Distinguishing Formula

ϕ distingishes $X \subseteq C$ from $Y \subseteq C$
if $X \subseteq [\![\phi]\!]$ and $[\![\phi]\!] \cap Y = \emptyset$.



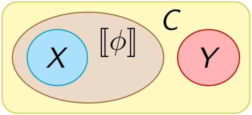


Certificates

For $c\colon C \to FC$, e.g. a transition system with states C

Distinguishing Formula

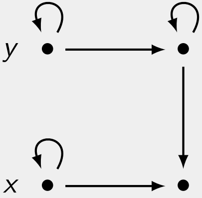
ϕ distingishes $X \subseteq C$ from $Y \subseteq C$
if $X \subseteq [\![\phi]\!]$ and $[\![\phi]\!] \cap Y = \emptyset$.



For $Y := C \setminus X$:

Certificate

ϕ certificate for $X \subseteq C$ if $[\![\phi]\!] = X$

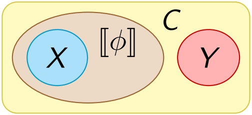


Certificates

For $c\colon C \to FC$, e.g. a transition system with states C

Distinguishing Formula

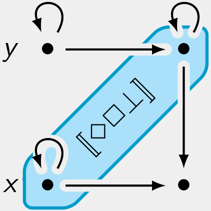
ϕ distingishes $X \subseteq C$ from $Y \subseteq C$
if $X \subseteq [\![\phi]\!]$ and $[\![\phi]\!] \cap Y = \emptyset$.



For $Y := C \setminus X$:

Certificate

ϕ certificate for $X \subseteq C$ if $[\![\phi]\!] = X$

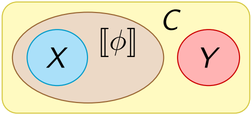


Certificates

For $c\colon C \to FC$, e.g. a transition system with states C

Distinguishing Formula

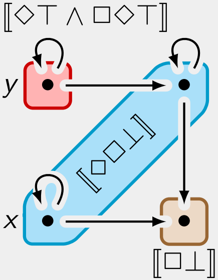
ϕ distingishes $X \subseteq C$ from $Y \subseteq C$
if $X \subseteq [\![\phi]\!]$ and $[\![\phi]\!] \cap Y = \emptyset$.



For $Y := C \setminus X$:

Certificate

ϕ certificate for $X \subseteq C$ if $[\![\phi]\!] = X$



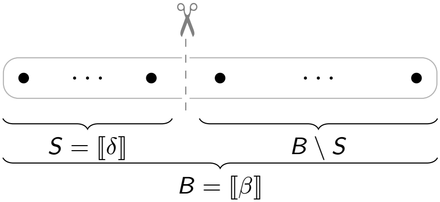
distinguishing formula for every pair $=$ certificates for behavioural equivalence classes

Initially

All states of $c\colon C \to FC$ are grouped w.r.t. $C \xrightarrow{c} FC \xrightarrow{F!} F1$

For example: live states $[\![\Diamond\top]\!]$ and deadlock states $[\![\Box\bot]\!]$.

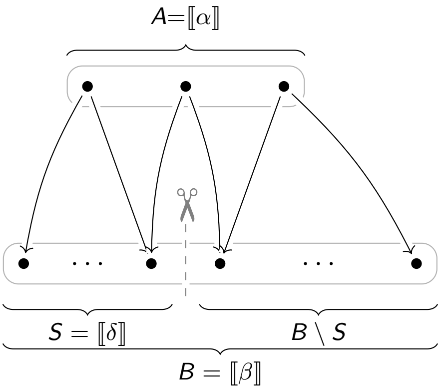
Refinement Step for $\mathcal{P}$



Initially

All states of $c\colon C \to FC$ are grouped w.r.t. $C \xrightarrow{c} FC \xrightarrow{F!} F1$
For example: live states $[\![\Diamond\top]\!]$ and deadlock states $[\![\Box\bot]\!]$.

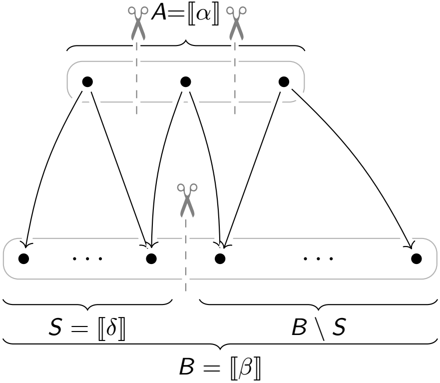
Refinement Step for $\mathcal{P}$



Initially

All states of $c\colon C \to FC$ are grouped w.r.t. $C \xrightarrow{c} FC \xrightarrow{F!} F1$
For example: live states $[\![\Diamond\top]\!]$ and deadlock states $[\![\Box\bot]\!]$.

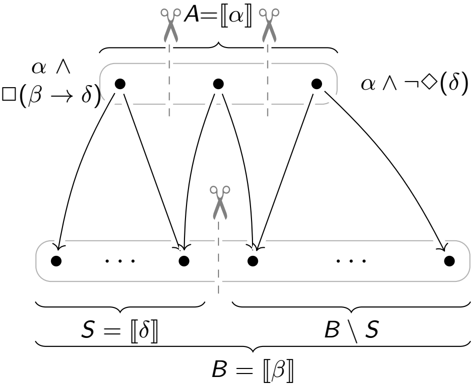
Refinement Step for $\mathcal{P}$



Initially

All states of $c\colon C \to FC$ are grouped w.r.t. $C \xrightarrow{c} FC \xrightarrow{F!} F1$
For example: live states $[\![\Diamond\top]\!]$ and deadlock states $[\![\Box\bot]\!]$.

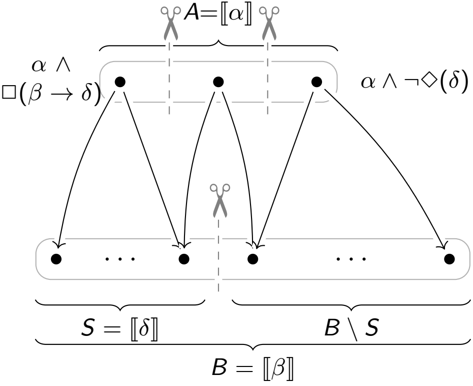
Refinement Step for $\mathcal{P}$



Initially

All states of $c\colon C \to FC$ are grouped w.r.t. $C \xrightarrow{c} FC \xrightarrow{F!} F1$
For example: live states $[\![\Diamond\top]\!]$ and deadlock states $[\![\Box\bot]\!]$.

Refinement Step for $\mathcal{P}$



States $x_1, x_2 \in A$ stay together iff

$$\mathcal{P}\chi_S^B(c(x_1)) = \mathcal{P}\chi_S^B(c(x_2)).$$

$$\chi_S^B\colon C \to 3$$

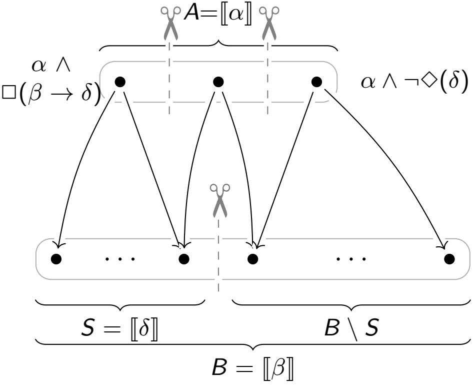
$$\chi_S^B(x) = \begin{cases} 2 & \text{if } x \in S \\ 1 & \text{if } x \in B \setminus S \\ 0 & \text{if } x \notin B \end{cases}$$

$$C \xrightarrow{c} \mathcal{P}C \xrightarrow{\mathcal{P}\chi_S^B} \mathcal{P}3$$

Initially

All states of $c\colon C \to FC$ are grouped w.r.t. $C \xrightarrow{c} FC \xrightarrow{F!} F1$
For example: live states $[\![\Diamond\top]\!]$ and deadlock states $[\![\Box\bot]\!]$.

Refinement Step for F



States $x_1, x_2 \in A$ stay together iff

$$F\chi_S^B(c(x_1)) = F\chi_S^B(c(x_2)).$$

$$\chi_S^B\colon C \to 3$$

$$\chi_S^B(x) = \begin{cases} 2 & \text{if } x \in S \\ 1 & \text{if } x \in B \setminus S \\ 0 & \text{if } x \notin B \end{cases}$$

$$C \xrightarrow{c} FC \xrightarrow{F\chi_S^B} F3$$

Generic Certificates

In the refinement step $S \subsetneq B \subseteq C$, we refine blocks w.r.t.

$$C \xrightarrow{c} FC \xrightarrow{F\chi_S^B} F3$$

Definition: $F3$-modalities

For $F\colon \mathrm{Set} \to \mathrm{Set}$, define the signature

$$\Lambda = \{\ulcorner t \urcorner/2 \mid t \in F3\} \qquad (\ulcorner t \urcorner = \text{syntactic representation})$$

Generic Certificates

In the refinement step $S \subsetneq B \subseteq C$, we refine blocks w.r.t.

$$C \xrightarrow{c} FC \xrightarrow{F\chi_S^B} F3$$

Definition: $F3$-modalities

For $F\colon \mathsf{Set} \to \mathsf{Set}$, define the signature

$$\Lambda = \{\ulcorner t \urcorner/2 \mid t \in F3\} \qquad (\ulcorner t \urcorner = \text{syntactic representation})$$

Semantics

$$\lambda_X^{\ulcorner t \urcorner}\colon (2^X)^2 \to 2^{FX} \qquad \lambda_X^{\ulcorner t \urcorner}(S, B) = \{t' \in FX \mid F\chi_{S\cap B}^B(t') = t\}$$

$$x \in [\![\ulcorner t \urcorner(\delta, \beta)]\!] \iff F\chi_{[\![\delta]\!]}^{[\![\beta]\!]}(c(x)) = t \qquad \text{if } [\![\delta]\!] \subseteq [\![\beta]\!]$$

Generic Certificates

In the refinement step $S \subsetneq B \subseteq C$, we refine blocks w.r.t.

$$C \xrightarrow{c} FC \xrightarrow{F\chi_S^B} F3$$

Definition: $F3$-modalities

For $F\colon \mathsf{Set} \to \mathsf{Set}$, define the signature

$$\Lambda = \{\ulcorner t \urcorner/2 \mid t \in F3\} \qquad (\ulcorner t \urcorner = \text{syntactic representation})$$

Semantics

$$\lambda_X^{\ulcorner t \urcorner}\colon (2^X)^2 \to 2^{FX} \qquad \lambda_X^{\ulcorner t \urcorner}(S,B) = \{t' \in FX \mid F\chi_{S\cap B}^B(t') = t\}$$

$$x \in [\![\ulcorner t \urcorner(\delta,\beta)]\!] \iff F\chi_{[\![\delta]\!]}^{[\![\beta]\!]}(c(x)) = t \qquad \text{if } [\![\delta]\!] \subseteq [\![\beta]\!]$$

Example for $F = \mathcal{P}$

$\ulcorner\{0,2\}\urcorner(\delta,\beta)$ is equivalent to $\overbrace{\Diamond\neg\beta}^{\text{‘0’}} \wedge \neg\overbrace{\Diamond(\beta \wedge \neg\delta)}^{\text{‘1’}} \wedge \overbrace{\Diamond(\delta \wedge \beta)}^{\text{‘2’}}$.

Coalgebraic Partition Refinement with Certificates

Algorithm: during partition refinement for $c\colon C \to FC$

1. Initially: annotate the block of x with the constant:
   $\ulcorner F!(c(x)) \urcorner$ where $t \in F1 \subseteq F3$.

Coalgebraic Partition Refinement with Certificates

Algorithm: during partition refinement for $c\colon C \to FC$

1. Initially: annotate the block of x with the constant: $\ulcorner F!(c(x)) \urcorner$ where $t \in F1 \subseteq F3$.
2. Whenever $A = [\![\alpha]\!]$ is split for $S \subsetneq B \subseteq C$, $S = [\![\delta]\!]$, $B = [\![\beta]\!]$: annotate the block of $x \in A$ with: $\alpha \wedge \ulcorner F\chi_S^B(c(x)) \urcorner(\delta, \beta)$

Coalgebraic Partition Refinement with Certificates

Algorithm: during partition refinement for $c\colon C \to FC$

1. Initially: annotate the block of x with the constant:
   $\ulcorner F!(c(x)) \urcorner$ where $t \in F1 \subseteq F3$.
2. Whenever $A = [\![\alpha]\!]$ is split for $S \subsetneq B \subseteq C$, $S = [\![\delta]\!]$, $B = [\![\beta]\!]$:
   annotate the block of $x \in A$ with: $\alpha \wedge \ulcorner F\chi_S^B(c(x)) \urcorner(\delta, \beta)$

Theorem.

If F is zippable [Concur'17] and (C, c) has n states and m transitions, then the algorithm...

1. is correct: every block is annotated with its certificate
2. constructs formulae of dag size: $\mathcal{O}(m \cdot \log n + n)$.
3. requires only constant run-time overhead.

Coalgebraic Partition Refinement

Theorem Wißmann, Dorsch, Milius, Schröder '20

If F is zippable and has a refinement interface, then coalgebraic partition refinement runs in

$$\mathcal{O}((m + n) \cdot \log n)$$

time for m transitions and n states.

Coalgebraic Partition Refinement

Theorem — Wißmann, Dorsch, Milius, Schröder '20

If F is zippable and has a refinement interface, then coalgebraic partition refinement runs in

$$\mathcal{O}((m+n)\cdot\log n)$$

time for m transitions and n states.

Corollary

Under these assumptions, we obtain certificates and distinguishing formulae in $\mathcal{O}((m+n)\cdot\log n)$ time.

Coalgebraic Partition Refinement

Theorem Wißmann, Dorsch, Milius, Schröder '20

If F is zippable and has a refinement interface, then coalgebraic partition refinement runs in

$$\mathcal{O}((m+n)\cdot\log n)$$

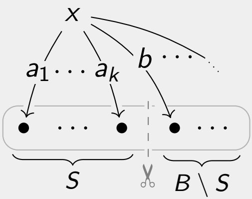
time for m transitions and n states.

Corollary

Under these assumptions, we obtain certificates and distinguishing formulae in $\mathcal{O}((m+n)\cdot\log n)$ time.

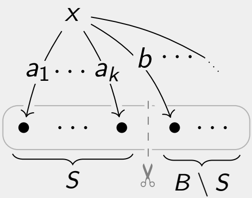
Examples

Automata, Tree Automata, Transition Systems, Weighted Systems, Markov chains, $\cdots$



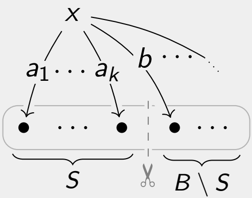
If you know weights $x \to S$ and $x \to B$, do you know the weight $x \to B \setminus S$?

If you know weights $x \to S$ and $x \to B$,
do you know the weight $x \to B \setminus S$?



Definition: Functor $F\colon \mathsf{Set} \to \mathsf{Set}$ is **cancellative** if

$$\langle F\chi_{\{1,2\}}, F\chi_{\{2\}}\rangle\colon F3 \to F2 \times F2 \quad \text{is injective}$$



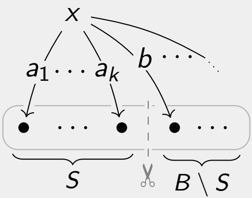
If you know weights $x \to S$ and $x \to B$,
do you know the weight $x \to B \setminus S$?

Definition: Functor $F\colon \mathsf{Set} \to \mathsf{Set}$ is **cancellative** if

$$\langle F\chi_{\{1,2\}}, F\chi_{\{2\}}\rangle\colon F3 \to F2 \times F2 \quad \text{is injective}$$

- Monoid-valued functor $M^{(-)}$ is cancellative
  $\Leftrightarrow$ the monoid M is cancellative

If you know weights $x \to S$ and $x \to B$,
do you know the weight $x \to B \setminus S$?

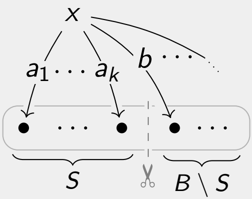


Definition: Functor $F\colon \mathsf{Set} \to \mathsf{Set}$ is **cancellative** if

$$\langle F\chi_{\{1,2\}}, F\chi_{\{2\}}\rangle\colon F3 \to F2 \times F2 \quad \text{is injective}$$

- Monoid-valued functor $M^{(-)}$ is cancellative
  $\Leftrightarrow$ the monoid M is cancellative
- e.g. $\mathbb{R}^{(-)}$ is cancellative, $\mathcal{P}_\mathrm{f} = 2^{(-)}$ is not.

If you know weights $x \to S$ and $x \to B$,
do you know the weight $x \to B \setminus S$?

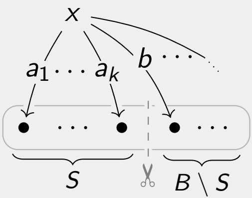


Definition: Functor $F\colon \mathrm{Set} \to \mathrm{Set}$ is **cancellative** if

$$\langle F\chi_{\{1,2\}}, F\chi_{\{2\}}\rangle \colon F3 \to F2 \times F2 \quad \text{is injective}$$

- Monoid-valued functor $M^{(-)}$ is cancellative
  $\Leftrightarrow$ the monoid M is cancellative
- e.g. $\mathbb{R}^{(-)}$ is cancellative, $\mathcal{P}_\mathrm{f} = 2^{(-)}$ is not.
- Identity and constant functors are cancellative.
- Cancellative functors closed under subfunctors, $+$, $\times$

If you know weights $x \to S$ and $x \to B$,
do you know the weight $x \to B \setminus S$?

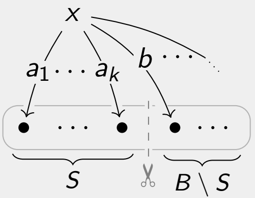


Definition: Functor $F\colon \mathrm{Set} \to \mathrm{Set}$ is **cancellative** if

$$\langle F\chi_{\{1,2\}}, F\chi_{\{2\}}\rangle\colon F3 \to F2 \times F2 \quad \text{is injective}$$

- Monoid-valued functor $M^{(-)}$ is cancellative
  $\Leftrightarrow$ the monoid M is cancellative
- e.g. $\mathbb{R}^{(-)}$ is cancellative, $\mathcal{P}_{\mathrm{f}} = 2^{(-)}$ is not.
- Identity and constant functors are cancellative.
- Cancellative functors closed under subfunctors, $+$, $\times$
- zippable and cancellative are independent.

If you know weights $x \to S$ and $x \to B$,
do you know the weight $x \to B \setminus S$?



Definition: Functor $F\colon \mathsf{Set} \to \mathsf{Set}$ is **cancellative** if

$$\langle F\chi_{\{1,2\}}, F\chi_{\{2\}}\rangle\colon F3 \to F2 \times F2 \quad \text{is injective}$$

- Monoid-valued functor $M^{(-)}$ is cancellative
  $\Leftrightarrow$ the monoid M is cancellative
- e.g. $\mathbb{R}^{(-)}$ is cancellative, $\mathcal{P}_{\mathrm{f}} = 2^{(-)}$ is not.
- Identity and constant functors are cancellative.
- Cancellative functors closed under subfunctors, $+$, $\times$
- zippable and cancellative are independent.

Theorem: If the functor F is cancellative, then we can replace

$$\ulcorner F\chi_S^B(c(x))\urcorner(\delta, \beta) \quad \text{with} \quad \ulcorner F\chi_S(c(x))\urcorner(\delta) \qquad (F2 \subseteq F3)$$

Conclusions

- Certificates for behavioural equivalence classes can be constructed efficiently, with only constant run-time overhead
- The size of the DAG representing the formula bounded by $\mathcal{O}(m \cdot \log n)$

Conclusions

- Certificates for behavioural equivalence classes can be constructed efficiently, with only constant run-time overhead
- The size of the DAG representing the formula bounded by $\mathcal{O}(m \cdot \log n)$

Also in the paper

Rewrite certificates to domain-specific modalities (e.g. $\Diamond$, $\langle a \rangle_p$).

Conclusions

- Certificates for behavioural equivalence classes can be constructed efficiently, with only constant run-time overhead
- The size of the DAG representing the formula bounded by $\mathcal{O}(m \cdot \log n)$

Also in the paper

Rewrite certificates to domain-specific modalities (e.g. $\Diamond$, $\langle a \rangle_p$).

Hennessy-Milner-Theorems for free

- for $\top$, $\wedge$, $\neg$, $\ulcorner t \urcorner/2$ ($t \in F3$) if F is zippable.
- for $\top$, $\wedge$, $\ulcorner t \urcorner/1$ ($t \in F2$) if F is zippable and cancellative.

References

[Pat03] Dirk Pattinson. “Coalgebraic modal logic: soundness, completeness and decidability of local consequence”. In: **Theoretical Computer Science** 309.1 (2003), pp. 177–193. DOI: https://doi.org/10.1016/S0304-3975(03)00201-9.

[WDMS20] Thorsten Wißmann, Ulrich Dorsch, Stefan Milius, Lutz Schröder. “Efficient and Modular Coalgebraic Partition Refinement”. In: **Logical Methods in Computer Science** Volume 16, Issue 1 (Jan. 2020). DOI: 10.23638/LMCS-16(1:8)2020. URL: https://lmcs.episciences.org/6064.