

# Generic and Efficient Partition Refinement

Thorsten Wißmann

University Erlangen-Nürnberg

Joint work with:

Hans-Peter Deifel, Ulrich Dorsch, Stefan Milius, Lutz Schröder

- Published in Concur 2017
- Extended version in LMCS 2020
- Implementation & more functors in FM2019

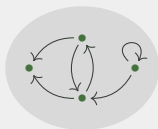
D-CON, March 12, 2020

# Generic and Efficient Partition Refinement

# Generic and Efficient Partition Refinement

## Coalgebras:

State based  
systems

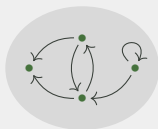


Labels, Non-Determinism,  
Probabilities, Automata, ...

# Generic and Efficient Partition Refinement

## Coalgebras:

State based  
systems



Labels, Non-Determinism,  
Probabilities, Automata, ...

## Modularity:

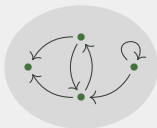
Combine  
system  
types by

$\circ$ ,  $\times$ ,  $+$

# Generic and Efficient Partition Refinement

## Coalgebras:

State based  
systems



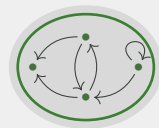
Labels, Non-Determinism,  
Probabilities, Automata, ...

## Modularity:

Combine  
system  
types by  
 $\circ, \times, +$

## Partition Refinement:

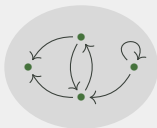
Successively distinguish  
different behaviour



# Generic and Efficient Partition Refinement

## Coalgebras:

State based  
systems



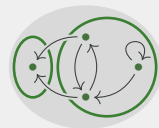
Labels, Non-Determinism,  
Probabilities, Automata, ...

## Modularity:

Combine  
system  
types by  
 $\circ$ ,  $\times$ ,  $+$

## Partition Refinement:

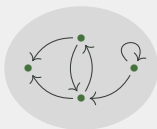
Successively distinguish  
different behaviour



# Generic and Efficient Partition Refinement

## Coalgebras:

State based  
systems



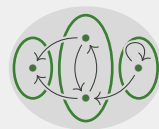
Labels, Non-Determinism,  
Probabilities, Automata, ...

## Modularity:

Combine  
system  
types by  
 $\circ$ ,  $\times$ ,  $+$

## Partition Refinement:

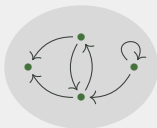
Successively distinguish  
different behaviour



# Generic and Efficient Partition Refinement

## Coalgebras:

State based  
systems



Labels, Non-Determinism,  
Probabilities, Automata, ...

## Modularity:

Combine  
system  
types by  
 $\circ, \times, +$

## Efficiency:

Complexity  
Analysis

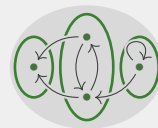
$$\mathcal{O}(m \cdot \log n)$$

Edges

States

## Partition Refinement:

Successively distinguish  
different behaviour







Share Common  
Structure & Ideas

Similar  
Run-Time

Variations in  
Details

## Share Common Structure & Ideas

Deterministic  
Finite Automata

$n \cdot \log n$  |  $|A| \cdot n \cdot \log n$   
Hopcroft '71 | Gries '73  
Knuutila '01

## Similar Run-Time

## Variations in Details

## Share Common Structure & Ideas

Deterministic  
Finite Automata

$$n \cdot \log n \quad |A| \cdot n \cdot \log n$$

Hopcroft '71    Gries '73  
                  Knuutila '01

(Labelled)

Transition Systems

$$m \cdot \log n$$

Paige, Tarjan '87  
Valmari '09

## Similar Run-Time

## Variations in Details

## Share Common Structure & Ideas

Deterministic  
Finite Automata

$n \cdot \log n$     $|A| \cdot n \cdot \log n$   
Hopcroft '71   Gries '73  
Knuutila '01

(Labelled)  
Transition Systems

$m \cdot \log n$   
Paige, Tarjan '87  
Valmari '09

Weighted Systems  
"Markov Chain Lumping"

$m \cdot \log n$   
Valmari, Franceschinis '10

## Similar Run-Time

## Variations in Details

## Share Common Structure & Ideas

Deterministic  
Finite Automata

$n \cdot \log n$  |  $|A| \cdot n \cdot \log n$   
Hopcroft '71 | Gries '73  
Knuutila '01

(Labelled)  
Transition Systems

$m \cdot \log n$   
Paige, Tarjan '87  
Valmari '09

Weighted Systems  
"Markov Chain Lumping"

$m \cdot \log n$   
Valmari, Franceschinis '10

## Similar Run-Time

## Variations in Details

Segala Systems

$m_{\text{dist}} \cdot \log m_{\text{acts}}$   
Groote, Verduzco,  
de Vink '18

## Share Common Structure & Ideas

Deterministic  
Finite Automata

$n \cdot \log n$  |  $|A| \cdot n \cdot \log n$   
Hopcroft '71 | Gries '73  
Knuutila '01

(Labelled)  
Transition Systems

$m \cdot \log n$   
Paige, Tarjan '87  
Valmari '09

Weighted Systems  
"Markov Chain Lumping"

$m \cdot \log n$   
Valmari, Franceschinis '10

## Similar Run-Time

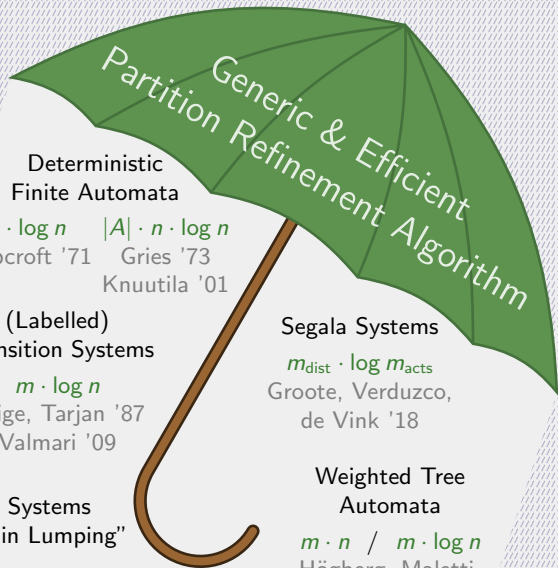
## Variations in Details

Segala Systems

$m_{\text{dist}} \cdot \log m_{\text{acts}}$   
Groote, Verduzco,  
de Vink '18

Weighted Tree  
Automata

$m \cdot n$  /  $m \cdot \log n$   
Högberg, Maletti,  
May '07



Deterministic  
Finite Automata

$n \cdot \log n$  |  $|A| \cdot n \cdot \log n$   
Hopcroft '71 | Gries '73  
Knuutila '01

(Labelled)  
Transition Systems

$m \cdot \log n$   
Paige, Tarjan '87  
Valmari '09

Weighted Systems  
"Markov Chain Lumping"

$m \cdot \log n$   
Valmari, Franceschinis '10

Segala Systems

$m_{\text{dist}} \cdot \log m_{\text{acts}}$   
Groote, Verduzco,  
de Vink '18

Weighted Tree  
Automata

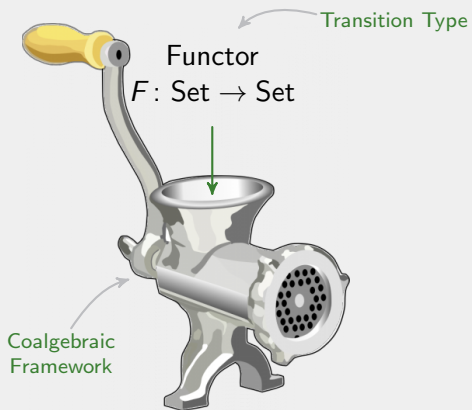
$m \cdot n / m \cdot \log n$   
Högberg, Maletti,  
May '07

# Coalgebra – Generic state based systems

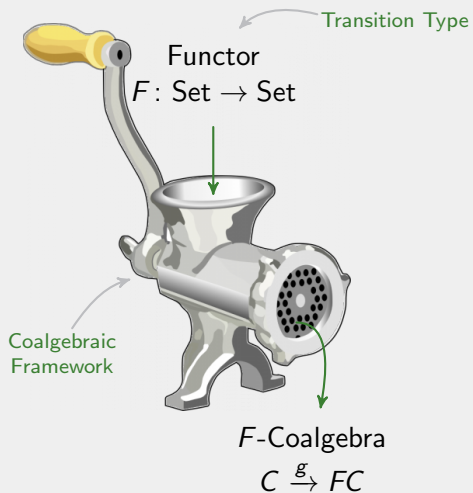




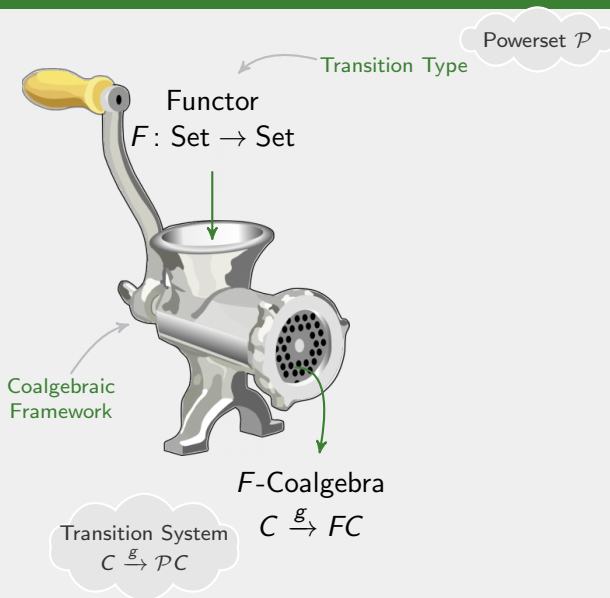
# Coalgebra – Generic state based systems



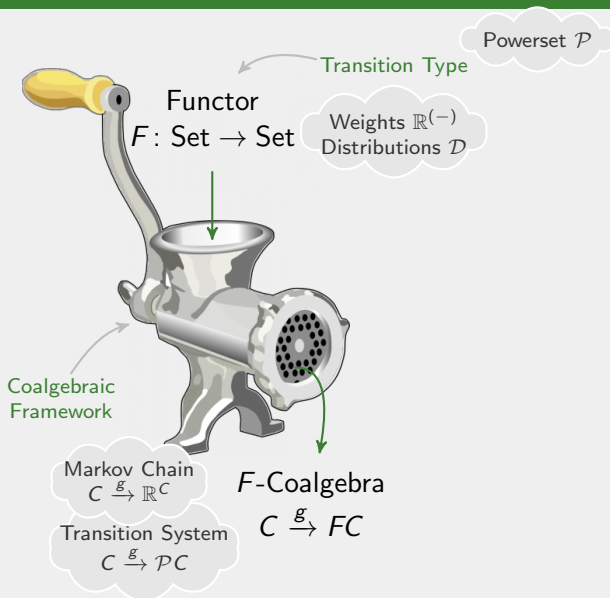
# Coalgebra – Generic state based systems



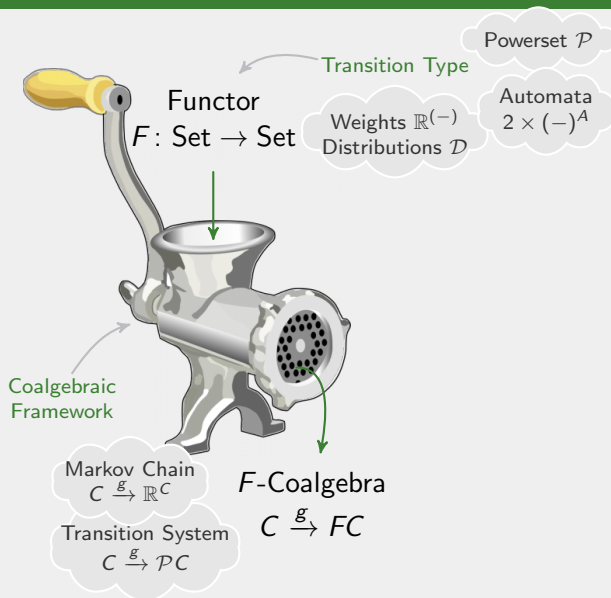
# Coalgebra – Generic state based systems



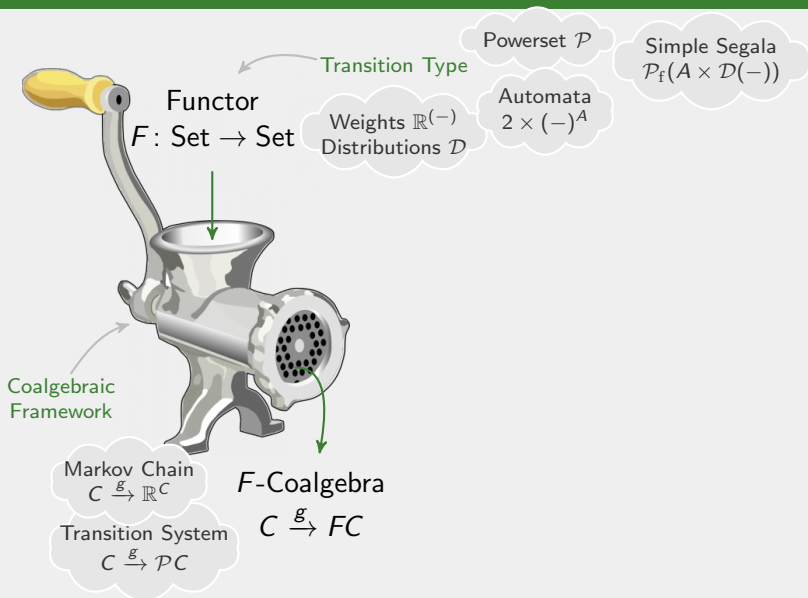
# Coalgebra – Generic state based systems



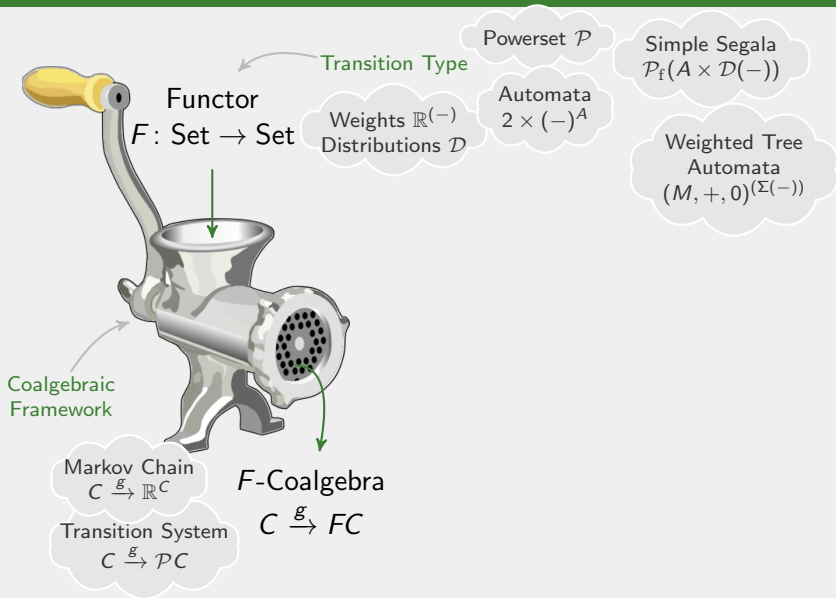
# Coalgebra – Generic state based systems



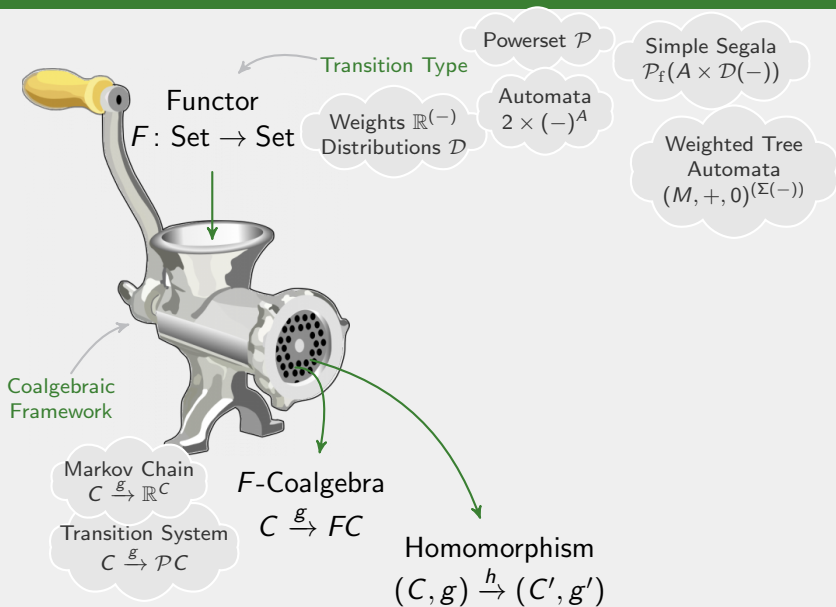
# Coalgebra – Generic state based systems



# Coalgebra – Generic state based systems

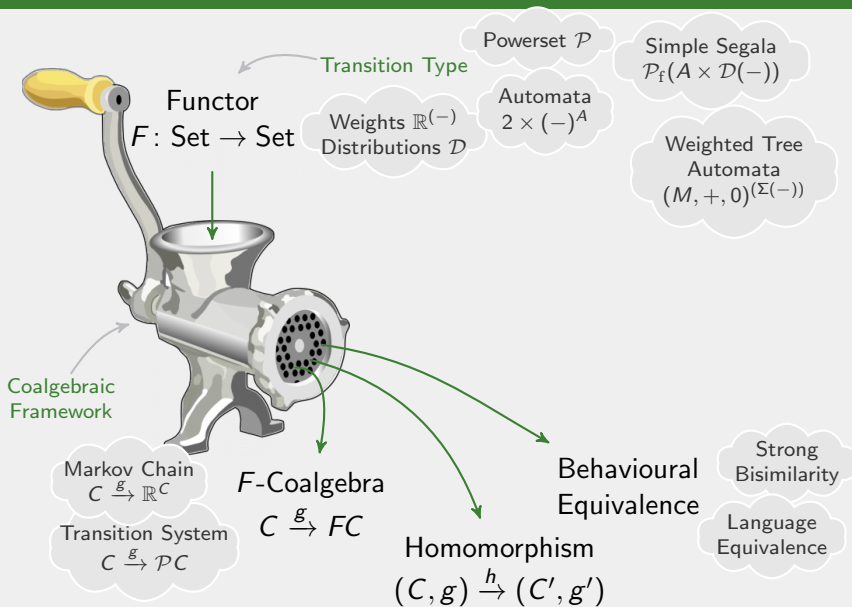


# Coalgebra – Generic state based systems

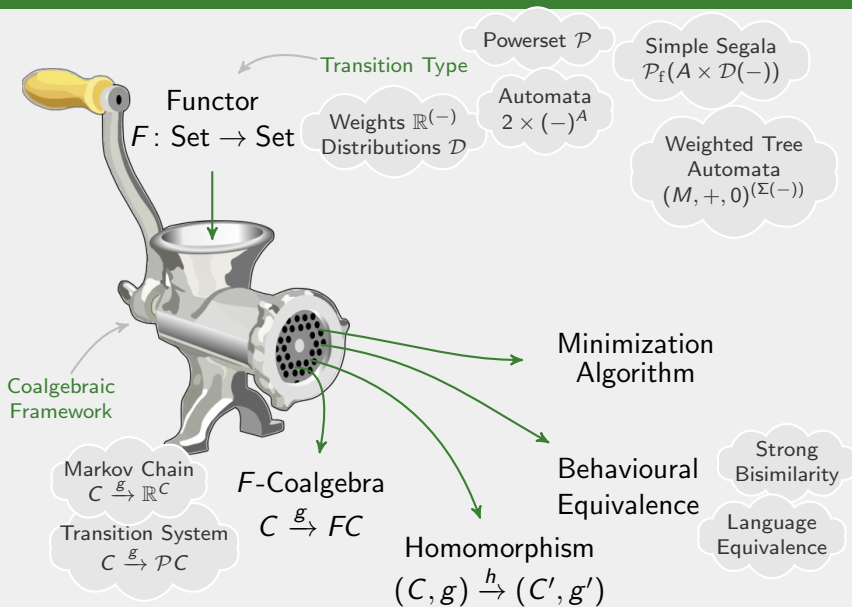




# Coalgebra – Generic state based systems



# Coalgebra – Generic state based systems



# The Coalgebraic Task

For a functor  $F : \text{Set} \rightarrow \text{Set}$

Given a coalgebra

$$\begin{array}{ccc}
 C & \xrightarrow{g} & FC \\
 h \downarrow & & \downarrow Fh \\
 C' & \xrightarrow{g'} & FC'
 \end{array}$$

no proper  
quotient

find the simple quotient

all equivalent  
behaviours  
identified

# The Coalgebraic Task

For a functor  $F : \text{Set} \rightarrow \text{Set}$

Given a coalgebra

$$\begin{array}{ccc}
 C & \xrightarrow{g} & FC \\
 h \downarrow & & \downarrow Fh \\
 C' & \xrightarrow{g'} & FC'
 \end{array}$$

no proper  
quotient

find the simple quotient

all equivalent  
behaviours  
identified

Instance

For  $2 \times (-)^A : \text{Set}$

Automata  
minimization

# The Coalgebraic Task

For a functor  $F : \text{Set} \rightarrow \text{Set}$

Given a coalgebra

$$\begin{array}{ccc} C & \xrightarrow{g} & FC \\ h \downarrow & & \downarrow Fh \\ C' & \xrightarrow{g'} & FC' \end{array}$$

no proper  
quotient

find the simple quotient

all equivalent  
behaviours  
identified

Instance

Instance

For  $2 \times (-)^A : \text{Set}$

Automata  
minimization

For  $\mathcal{P}_f : \text{Set}$

Bisimilarity  
minimization

# The Coalgebraic Task

For a functor  $F : \text{Set} \rightarrow \text{Set}$

Given a coalgebra

$$\begin{array}{ccc} C & \xrightarrow{g} & FC \\ h \downarrow & & \downarrow Fh \\ C' & \xrightarrow{g'} & FC' \end{array}$$

no proper  
quotient

find the simple quotient

all equivalent  
behaviours  
identified

Instance

Instance

Instance

For  $2 \times (-)^A : \text{Set}$

Automata  
minimization

For  $\mathcal{P}_f : \text{Set}$

Bisimilarity  
minimization

For  $\mathbb{R}^{(-)} : \text{Set}$

Markov chain  
lumping

# The Coalgebraic Task

For a functor  $F : \text{Set} \rightarrow \text{Set}$

Given a coalgebra

$$\begin{array}{ccc} C & \xrightarrow{g} & FC \\ h \downarrow & & \downarrow Fh \\ C' & \xrightarrow{g'} & FC' \end{array}$$

no proper  
quotient

find the simple quotient

all equivalent  
behaviours  
identified

Instance

Instance

Instance

Instance

For  $2 \times (-)^A : \text{Set}$

Automata  
minimization

For  $\mathcal{P}_f : \text{Set}$

Bisimilarity  
minimization

For  $\mathbb{R}^{(-)} : \text{Set}$

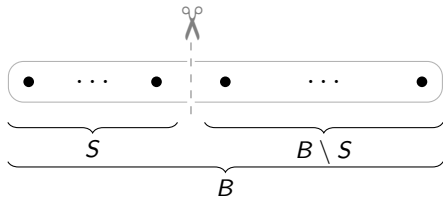
Markov chain  
lumping

...

## Initially

All states of  $g: C \rightarrow FC$  are grouped w.r.t.  $C \xrightarrow{g} FC \xrightarrow{F!} F1$   
(e.g. final vs. non-final states)

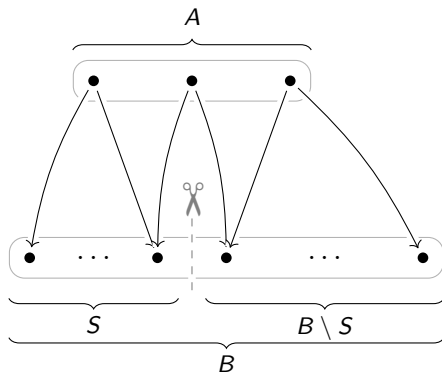
## Refinement Step for $\mathcal{P}$





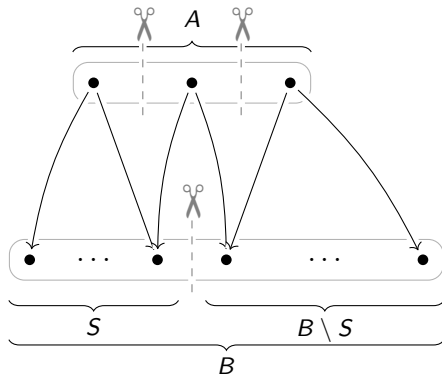
## Initially

All states of  $g: C \rightarrow FC$  are grouped w.r.t.  $C \xrightarrow{g} FC \xrightarrow{F!} F1$   
 (e.g. final vs. non-final states)

Refinement Step for  $\mathcal{P}$ 

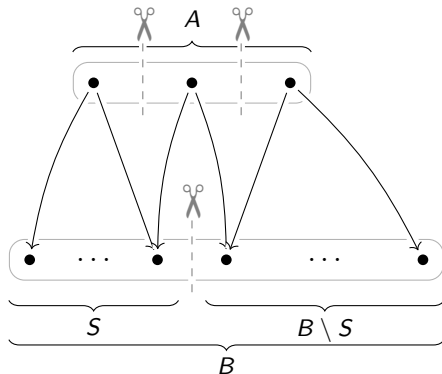
## Initially

All states of  $g: C \rightarrow FC$  are grouped w.r.t.  $C \xrightarrow{g} FC \xrightarrow{F!} F1$   
 (e.g. final vs. non-final states)

Refinement Step for  $\mathcal{P}$ 

## Initially

All states of  $g: C \rightarrow FC$  are grouped w.r.t.  $C \xrightarrow{g} FC \xrightarrow{F^!} F1$   
(e.g. final vs. non-final states)

Refinement Step for  $\mathcal{P}$ 

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

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

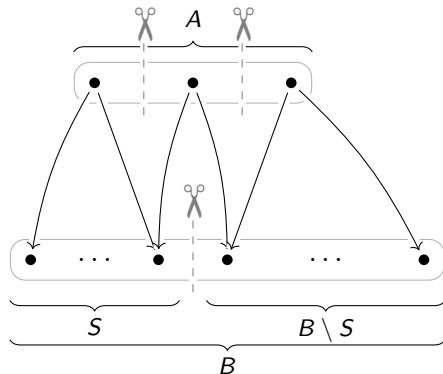
$$\chi_S^B: C \rightarrow 3$$

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

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

## Initially

All states of  $g: C \rightarrow FC$  are grouped w.r.t.  $C \xrightarrow{g} FC \xrightarrow{F^1} F1$   
(e.g. final vs. non-final states)

Refinement Step for  $F$ 

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

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

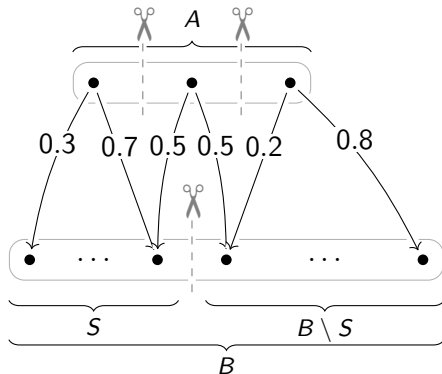
$$\chi_S^B: C \rightarrow 3$$

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

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

## Initially

All states of  $g: C \rightarrow FC$  are grouped w.r.t.  $C \xrightarrow{g} FC \xrightarrow{F^1} F1$   
(e.g. final vs. non-final states)

Refinement Step for  $F$ 

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

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

$$\chi_S^B: C \rightarrow 3$$

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

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

How to compute  $C \xrightarrow{c} FC \xrightarrow{F\chi_S^B} F3$  efficiently?

## Functor Encoding

Labels  $A$

$$b : FX \rightarrow \mathcal{B}(A \times X)$$

Bags

How to compute  $C \xrightarrow{c} FC \xrightarrow{F\chi_S^B} F3$  efficiently?

## Functor Encoding

Labels  $A$

$$b : FX \rightarrow \mathcal{B}(A \times X)$$

Bags

## Refinement Interface

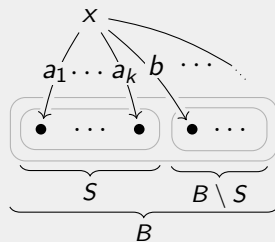
Type  $W$  (abstract, could be ints, reals, trees, ...)

$$\text{init} : F1 \times \mathcal{B}A \rightarrow W$$

$$\text{update} : \mathcal{B}A \times W \rightarrow W \times F3 \times W$$

Labels to  $S$

Weight of  $B$



How to compute  $C \xrightarrow{c} FC \xrightarrow{F\chi_S^B} F3$  efficiently?

## Functor Encoding

Labels  $A$

$$b : FX \rightarrow \mathcal{B}(A \times X)$$

Bags

## Refinement Interface

Type  $W$  (abstract, could be ints, reals, trees, ...)

$$\text{init} : F1 \times \mathcal{B}A \rightarrow W$$

$$\text{update} : \mathcal{B}A \times W \rightarrow W \times F3 \times W$$

Labels to  $S$

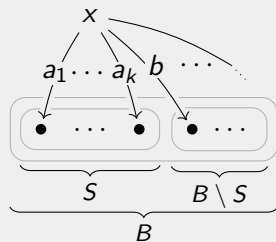
Weight of  $B$

Example:  $FX = \mathbb{R}^{(X)}$

$$A := \mathbb{R} \quad W := \mathbb{R} \times \mathbb{R}$$

$$\text{init}(\_, \ell) = (0, \Sigma \ell)$$

$$\text{update}(\ell, (r, b)) = ((r + b - \Sigma \ell, \Sigma \ell), \dots)$$





INITIALIZATION: Partitioning w.r.t.  $C \xrightarrow{c} FC \xrightarrow{F!} F1$

for  $e \in E$ ,  $e = x \xrightarrow{a} y$  do  
 add  $e$  to  $\text{toSub}[x]$  and  $\text{pred}[y]$

for  $x \in X$  do  
 $\rho_X :=$  new cell in  $\text{deref}$  containing  $\text{init}(\text{type}[x], \mathcal{B}(\pi_2 \cdot \text{graph})(\text{toSub}[x]))$   
 for  $e \in \text{toSub}[x]$  do  $\text{lastW}[e] = \rho_X$   
 $\text{toSub}[x] := \emptyset$

$X/P :=$  group  $X$  by  $\text{type}: X \rightarrow F1$ .

REFINEMENT STEP: Refine by  $C \xrightarrow{c} F3 \xrightarrow{F\chi_S^T} F3$

SPLIT( $X/P, S$ )

$M := \emptyset \subseteq X/P \times F3$

for  $y \in S$ ,  $e \in \text{pred}[y]$  do

$x \xrightarrow{a} y := e$

$B :=$  block with  $x \in B \in X/P$

if  $\text{mark}_B$  is empty then

$w_T^x := \text{deref} \cdot \text{lastW}[e]$

$v_\emptyset := \pi_2 \cdot \text{update}(\emptyset, w_T^x)$

add  $(B, v_\emptyset)$  to  $M$

if  $\text{toSub}[x] = \emptyset$  then

add  $(x, \text{lastW}[e])$  to  $\text{mark}_B$

add  $e$  to  $\text{toSub}[x]$

for  $(B, v_\emptyset) \in M$  do

$B_{\neq \emptyset} := \emptyset \subseteq X \times F3$

for  $(x, \rho_C)$  in  $\text{mark}_B$  do

$\ell := \mathcal{B}(\pi_2 \cdot \text{graph})(\text{toSub}[x])$

$(w_S^x, v^x, w_C^x) := \text{update}(\ell, \text{deref}[\rho_T])$

$\text{deref}[\rho_T] := w_T^x \setminus S$

$\rho_S :=$  new cell containing  $w_S^x$

for  $e \in \text{toSub}[x]$  do  $\text{lastW}[e] := \rho_S$

$\text{toSub}[x] := \emptyset$

if  $v^x \neq v_\emptyset$  then

remove  $x$  from  $B$

insert  $(x, v^x)$  into  $B_{\neq \emptyset}$

$\text{mark}_B := \emptyset$

$B_1 \times \{v_1\}, \dots, B_\ell \times \{v_\ell\} :=$

group  $B_{\neq \emptyset}$  by  $\pi_2: X \times F3 \rightarrow F3$

insert  $B_1, \dots, B_\ell :=$  into  $X/P$

(a) Collecting predecessor blocks

(b) Splitting predecessor blocks

## Efficiency

$F: \text{Set} \rightarrow \text{Set}$  is zippable

&

$F$  has a refinement interface  
(with linear run-time)

$\implies$

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

Edges

States

Functor  $F$  zippable, if the canonical map

$$F(L + R) \longrightarrow F(L + 1) \times F(1 + R) \text{ is injective.}$$

E.g. Id, Constants,  $\times$ ,  $+$ ,  $\hookrightarrow$ ,  $M^{(-)}$ , part. additive  $\longleftarrow F(X + Y) \mapsto FX \times FY$

Functor  $F$  zipable, if the canonical map

$$F(L + R) \longrightarrow F(L + 1) \times F(1 + R) \text{ is injective.}$$

E.g. Id, Constants,  $\times$ ,  $+$ ,  $\hookrightarrow$ ,  $M^{(-)}$ , part. additive  $\longleftarrow F(X + Y) \mapsto FX \times FY$

Examples for sets  $L = \{a_1, a_2, a_3\}$ ,  $R = \{b_1, b_2\}$ ,  $1 = \{-\}$

$$\begin{array}{l} a_1 \ a_2 \ b_1 \ a_3 \ b_2 \xrightarrow{\text{unzip}} \\ (a_1 \ a_2 \ - \ a_3 \ -, \\ \ - \ - \ b_1 \ - \ b_2) \end{array}$$

$(-)^5$  is zipable

$$\begin{array}{l} \{a_1, a_2, b_1\} \xrightarrow{\text{unzip}} \\ (\{a_1, a_2, -\}, \\ \{-, b_1\}) \end{array}$$

$\mathcal{P}_f$  is zipable

Functor  $F$  zipable, if the canonical map

$$F(L + R) \longrightarrow F(L + 1) \times F(1 + R) \text{ is injective.}$$

E.g. Id, Constants,  $\times$ ,  $+$ ,  $\hookrightarrow$ ,  $M^{(-)}$ , part. additive  $\longleftarrow F(X + Y) \mapsto FX \times FY$

Examples for sets  $L = \{a_1, a_2, a_3\}$ ,  $R = \{b_1, b_2\}$ ,  $1 = \{-\}$

$$\begin{array}{l} a_1 \ a_2 \ b_1 \ a_3 \ b_2 \xrightarrow{\text{unzip}} \\ (a_1 a_2 - a_3 -, \\ \quad - - b_1 - b_2) \end{array}$$

$(-)^5$  is zipable

$$\begin{array}{l} \{a_1, a_2, b_1\} \xrightarrow{\text{unzip}} \\ (\{a_1, a_2, -\}, \\ \quad \{-, b_1\}) \end{array}$$

$\mathcal{P}_f$  is zipable

$$\{\{a_1, b_1\}, \{a_2, b_2\}\} \quad \{\{a_1, b_2\}, \{a_2, b_1\}\}$$

$$\begin{array}{c} \text{unzip} \lrcorner \longrightarrow (\{\{a_1, -\}, \{a_2, -\}\}, \{\{-, b_1\}, \{-, b_2\}\}) \longleftarrow \lrcorner \text{unzip} \end{array}$$

$\mathcal{P}_f \mathcal{P}_f$  is not zipable

~~Composition~~

~~Quotients~~

## Efficiency

$F: \text{Set} \rightarrow \text{Set}$  is zippable

&

$F$  has a refinement interface  
(with linear run-time)

$\implies$

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

Edges

States

## Efficiency

$F: \text{Set} \rightarrow \text{Set}$  is zippable  
 &  
 $F$  has a refinement interface  
 (with linear run-time)

$\implies$  Minimization runs  
 in  $\mathcal{O}((m+n) \cdot \log n)$

Edges                      States

## Refinement Interfaces for

- Polynomial Functors  $\Sigma$
- $G^{(-)}$ ,  $G$  abelian group, e.g.  $\mathbb{R}^{(-)}$ , finite multisets  $\mathcal{B} = \mathbb{N}^{(-)}$
- $\mathcal{P}_f$  finite powerset
- $M^{(-)}$ ,  $M$  commutative monoid (additional factor  $\log \min(|M|, m)$ )

System	Functor $FX$	Run-Time ( $m \geq n$ )		Specific algorithm
Transition Systems	$\mathcal{P}_f X$	$m \cdot \log n$	$=$	$m \cdot \log n$ Paige, Tarjan 1987
Markov Chains	$\mathbb{R}^X$	$m \cdot \log n$	$=$	$m \cdot \log n$ Valmari, Franceschinis 2010
DFA	$2 \times X^A$ ( $A$ fixed)	$n \cdot \log n$	$=$	$n \cdot \log n$ Hopcroft 1971
Colour Refinement	$\mathcal{B}X$	$m \cdot \log n$	$=$	$m \cdot \log n$ Berkholz, Bonsma, Grohe 2017



## The Tool CoPaR

- Implementation in Haskell
- Users can easily implement new refinement interfaces.
- Available at <https://gitlab.cs.fau.de/i8/copar>

## The Tool CoPaR

- Implementation in Haskell
- Users can easily implement new refinement interfaces.
- Available at <https://gitlab.cs.fau.de/i8/copar>

## Refinement Interface Type

Math:

$$\text{init}: F1 \times \mathcal{BA} \rightarrow W$$

$$\text{update}: \mathcal{BA} \times W \rightarrow W \times F3 \times W$$

Haskell:

```
class (Ord (F1 f), Ord (F3 f)) => RefinementInterface f where
  init  :: F1 f -> [Label f] -> Weight f
  update :: [Label f] -> Weight f -> (Weight f, F3 f, Weight f)
```

## Example: Refinement Interface Implementation for $\mathbb{R}^{(-)}$

**Math:**

$$\text{init}(f_1, e) = (0, \sum e)$$
$$\text{update}(e, (r, c)) = ((r + c - \sum e, \sum e), (r, c - \sum e, \sum e), (\sum e + r, c - \sum e))$$

**Haskell:**

```
instance RefinementInterface R where  
  init f1 e = (0, sum e)  
  update e (r, c) = ((r + c - sum e, sum e),  
                    (r, c - sum e, sum e),  
                    (sum e + r, c - sum e))
```

Example: Input coalgebra for  $FX = \mathbb{R}^{(X)}$

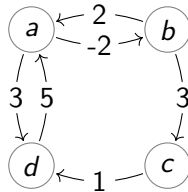
$\mathbb{R}^{(X)}$

a: { d: 3, b: -2 }

b: { a: 2, c: 3 }

c: { d: 1 }

d: { a: 5 }



Example: Input coalgebra for  $FX = \mathbb{R}^{(X)}$

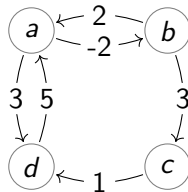
$\mathbb{R}^{(X)}$

a: { d: 3, b: -2 }

b: { a: 2, c: 3 }

c: { d: 1 }

d: { a: 5 }



Output

Block 0: d, b

Block 1: a, c

# Modularity: Composed System Types

$$FX = \mathcal{P}_f(\mathcal{D}(A \times X))$$

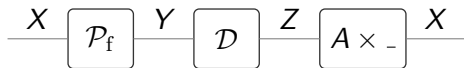


$$\Rightarrow H: \text{Set}^3 \rightarrow \text{Set}^3 \quad H(X, Y, Z) = (\mathcal{P}_f Y, \mathcal{D}Z, A \times X)$$

$$\Rightarrow H': \text{Set} \rightarrow \text{Set} \quad H'X = \mathcal{P}_f X + \mathcal{D}X + A \times X$$

# Modularity: Composed System Types

$$FX = \mathcal{P}_f(\mathcal{D}(A \times X))$$



$$\implies H: \text{Set}^3 \rightarrow \text{Set}^3 \quad H(X, Y, Z) = (\mathcal{P}_f Y, \mathcal{D}Z, A \times X)$$

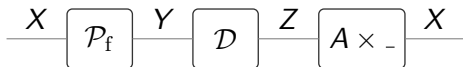
$$\implies H': \text{Set} \rightarrow \text{Set} \quad H'X = \mathcal{P}_f X + \mathcal{D}X + A \times X$$

Theorem (for every such  $F$ )

Every  $F$ -coalgebra can be transformed into a  $H'$ -coalgebra, and they have the same simple quotient.

# Modularity: Composed System Types

$$FX = \mathcal{P}_f(\mathcal{D}(A \times X))$$



$$\implies H: \text{Set}^3 \rightarrow \text{Set}^3 \quad H(X, Y, Z) = (\mathcal{P}_f Y, \mathcal{D}Z, A \times X)$$

$$\implies H': \text{Set} \rightarrow \text{Set} \quad H'X = \mathcal{P}_f X + \mathcal{D}X + A \times X$$

## Theorem (for every such $F$ )

Every  $F$ -coalgebra can be transformed into a  $H'$ -coalgebra, and they have the same simple quotient.

## Efficiency

For zippable functors  $F_1, \dots, F_n$  with refinement interfaces one can construct a refinement interface for  $F_1 + \dots + F_n$ .



# In CoPaR

## Modularity reduction during preprocessing <sup>monoid</sup>

- Implemented basic functors:  $\Sigma$ ,  $\mathcal{P}_f$ ,  $\mathcal{B}$ ,  $\mathcal{D}$ ,  $M^{(-)}$ ,  
 for  $M = \underbrace{\mathbb{N} \mid \mathbb{Q} \mid \mathbb{Z} \mid \mathbb{R}}_{\text{with } +} \mid (\mathbb{Z}, \max) \mid (\mathbb{R}, \max) \mid (\mathcal{P}_f(64), \cup)$

## In CoPaR

## Modularity reduction during preprocessing

- Implemented basic functors:  $\Sigma$ ,  $\mathcal{P}_f$ ,  $\mathcal{B}$ ,  $\mathcal{D}$ ,  $M^{(-)}$ ,  
for  $M = \mathbb{N} \mid \underbrace{\mathbb{Q} \mid \mathbb{Z} \mid \mathbb{R}}_{\text{with } +} \mid (\mathbb{Z}, \max) \mid (\mathbb{R}, \max) \mid (\mathcal{P}_f(64), \cup)$

- Interfaces for composed functors are automatically derived:

$$\begin{array}{l}
 \text{functor variable} \\
 \downarrow \\
 F ::= \mathbf{X} \mid \mathcal{P}_f F \mid \mathcal{B}F \mid \mathcal{D}F \mid M^{(F)} \mid \underbrace{N \mid F + F \mid F \times F \mid F^A}_{\text{polynomial constructs } \Sigma} \\
 N ::= \mathbb{N} \mid A \quad A ::= \{s_1, \dots, s_n\}
 \end{array}$$

System	Functor $FX$	Run-Time ( $m \geq n$ )		Specific algorithm
Transition Systems	$\mathcal{P}_f X$	$m \cdot \log n$	$=$	$m \cdot \log n$ Paige, Tarjan 1987
Markov Chains	$\mathbb{R}^{(X)}$	$m \cdot \log n$	$=$	$m \cdot \log n$ Valmari, Franceschinis 2010
DFA	$2 \times X^A$ ( $A$ fixed)	$n \cdot \log n$	$=$	$n \cdot \log n$ Hopcroft 1971
Colour Refinement	$\mathcal{B}X$	$m \cdot \log n$	$=$	$m \cdot \log n$ Berkholz, Bonsma, Grohe 2017

System	Functor $FX$	Run-Time ( $m \geq n$ )		Specific algorithm	
Transition Systems	$\mathcal{P}_f X$	$m \cdot \log n$	$=$	$m \cdot \log n$	Paige, Tarjan 1987
LTS	$\mathcal{P}_f(\mathbb{N} \times X)$	$m \cdot \log m$	$=$ $>$	$m \cdot \log m$ $m \cdot \log n$	Dovier, Piazza, Policriti 2004 Valmari 2009
Markov Chains	$\mathbb{R}^{(X)}$	$m \cdot \log n$	$=$	$m \cdot \log n$	Valmari, Franceschinis 2010
DFA	$2 \times X^A$ ( $A$ fixed)	$n \cdot \log n$	$=$	$n \cdot \log n$	Hopcroft 1971
	$2 \times \mathcal{P}_f(A \times X)$	$ A  \cdot n \cdot \log n$	$=$	$ A  \cdot n \cdot \log n$	Gries 1973/Knuutila 2001
Segala Systems	$\mathcal{P}_f(A \times \mathcal{D}X)$	$m_{\mathcal{D}} \cdot \log m_{\mathcal{P}_f}$	$<$ $=$	$m \cdot \log n$ $m_{\mathcal{D}} \cdot \log m_{\mathcal{P}_f}$	Baier, Engelen, Majster-Cederbaum 2000 Groote, Verduzco, de Vink 2018
Colour Refinement	$\mathcal{B}X$	$m \cdot \log n$	$=$	$m \cdot \log n$	Berkholz, Bonsma, Grohe 2017
Weighted Tree Automata	$M^{(\Sigma X)}$ $M$ non-cancellative	$m \cdot \log^2 m$	$<$	$m \cdot n$	Högberg, Maletti, May 2007
	$M^{(\Sigma X)}$ $M$ cancellative	$m \cdot \log m$	$=$ $\Sigma$ fixed	$m \cdot \log n$	Högberg, Maletti, May 2007

System	Functor $FX$	Run-Time ( $m \geq n$ )	Specific algorithm
Transition Systems	$\mathcal{P}_f X$	$m \cdot \log n$ =	$m \cdot \log n$ Paige, Tarjan 1987
LTS	$\mathcal{P}_f(\mathbb{N} \times X)$	$m \cdot \log m$ = $>$	$m \cdot \log m$ Dovier, Piazza, Policriti 2004 $m \cdot \log n$ Valmari 2009
Markov Chains	$\mathbb{R}(X)$	$m \cdot \log n$ =	$m \cdot \log n$ Valmari, Franceschinis 2010
DFA	$2 \times X^A$ ( $A$ fixed)	$n \cdot \log n$ =	$n \cdot \log n$ Hopcroft 1971
	$2 \times \mathcal{P}_f(A \times X)$	$ A  \cdot n \cdot \log n$ =	$ A  \cdot n \cdot \log n$ Gries 1973/Knuutila 2001
Segala Systems	$\mathcal{P}_f(A \times \mathcal{D}^X)$	$m_{\mathcal{D}} \cdot \log m_{\mathcal{P}_f}$ =	$m \cdot \log n$ Baier, Engelen, Majster-Cederbaum 2000 $m_{\mathcal{D}} \cdot \log m_{\mathcal{P}_f}$ Groote, Verduzco, de Vink 2018
Colour Refinement	$\mathcal{B}X$	$m \cdot \log n$ =	$m \cdot \log n$ Berkholz, Bonsma, Grohe 2017
Weighted Tree Automata	$M(\Sigma X)$ $M$ non-cancellative	$m \cdot \log^2 m$ <	$m \cdot n$ Högberg, Maletti, May 2007
	$M(\Sigma X)$ $M$ cancellative	$m \cdot \log m$ =	$m \cdot \log n$ Högberg, Maletti, May 2007

Generic & Efficient

$\Sigma$  fixed

System	Functor $FX$	Run-Time ( $m \geq n$ )	Specific algorithm
Transition Systems	$\mathcal{P}_f X$	$m \cdot \log n$ =	$m \cdot \log n$ Paige, Tarjan 1987
LTS	$\mathcal{P}_f(\mathbb{N} \times X)$	$m \cdot \log m$ = $>$	$m \cdot \log m$ Dovier, Piazza, Policriti 2004 $m \cdot \log n$ Valmari 2009
Markov Chains	$\mathbb{R}(X)$	$m \cdot \log n$ =	$m \cdot \log n$ Valmari, Franceschinis 2010
DFA	$2 \times X^A$ ( $A$ fixed)	$n \cdot \log n$ =	$n \cdot \log n$ Hopcroft 1971
	$2 \times \mathcal{P}_f(A \times X)$	$ A  \cdot n \cdot \log n$ =	$ A  \cdot n \cdot \log n$ Gries 1973/Knuutila 2001
Segala Systems	$\mathcal{P}_f(A \times \mathcal{D}^X)$	$m \cdot \log n$ = $m \cdot \log n$	$m \cdot \log n$ Baier, Engelen, Majster-Cederbaum 2000 $m \cdot \log n$ Groote, Verduzco, de Vink 2018
Colour Refinement	$\mathcal{B}X$	$m \cdot \log n$	$m \cdot \log n$ Giesma, Grohe 2017
Weighted Tree Automata	$M(\Sigma X)$ $M$ non-cancellative $M(\Sigma X)$ $M$ cancellative	$m \cdot \log^2 m$ < $m \cdot \log m$ =	$m \cdot \log^2 m$ Höfberg, Maletti, May 2007 $m \cdot \log m$ Höfberg, Maletti, May 2007
		$\Sigma$ fixed	

Generic & Efficient

Articles & Tool  
[tinyurl.com/coalgebra](http://tinyurl.com/coalgebra)

System	Functor $FX$	Run-Time ( $m \geq n$ )	Specific algorithm
Transition Systems	$\mathcal{P}_f X$	$m \cdot \log n$ =	$m \cdot \log n$ Paige, Tarjan 1987
LTS	$\mathcal{P}_f(\mathbb{N} \times X)$	$m \cdot \log m$	Dovier, Piazza, Policriti 2004
		$n \cdot \log n$	Valmari 2009
Markov Chains	$\mathbb{R}(X)$	$m \cdot \log n$	Valmari, Franceschinis 2010
DFA	$2 \times X^A$ ( $A$ fixed)	$n \cdot \log n$ =	$m \cdot \log n$ Hopcroft 1971
	$2 \times \mathcal{P}_f(A \times X)$	$ A  \cdot n \cdot \log n$ =	$ A  \cdot n \cdot \log n$ Gries 1973/Knuutila 2001
Segala Systems	$\mathcal{P}_f(A \times \mathcal{D}X)$	$m \cdot \log n$	Baier, Engelen, Majster-Cederbaum 2000
		$m \cdot \log m$ =	$m \cdot \log m$ Groote, Verduzco, de Vink 2018
Colour Refinement	$\mathcal{B}X$	$m \cdot \log n$	Alami, Grohe 2017
Weighted Tree Automata	$M(\Sigma X)$	$m \cdot \log^2 m$ <	Alami, Grohe, Maletti, May 2007
	$M$ non-cancellative		
	$M(\Sigma X)$	$m \cdot \log m$ =	$m \cdot \log m$ Högberg, Maletti, May 2007
	$M$ cancellative	$\Sigma$ fixed	

More instances:  
further system types  
& equivalences

Generic & Efficient

Articles & Tool  
[tinyurl.com/coalgebra](http://tinyurl.com/coalgebra)

System	Functor $FX$	Run-Time ( $m \geq n$ )	Specific algorithm
Transition Systems	$\mathcal{P}_f X$	$m \cdot \log n$ =	$m \cdot \log n$ Paige, Tarjan 1987
LTS	$\mathcal{P}_f(\mathbb{N} \times X)$	$m \cdot \log m$	Dovier, Piazza, Policriti 2004
Markov Chains	$\mathbb{R}(X)$	$m \cdot \log n$	Valmari 2009
DFA	$\mathcal{P}_f(A \times D^X)$ ( $A$ fixed)	$n \cdot \log n$ =	$m \cdot \log n$ Hopcroft 1971
	$X)$	$ A  \cdot n \cdot \log n$ =	$ A  \cdot n \cdot \log n$ Gries 1973/Knuutila 2001
Segala Systems	$\mathcal{P}_f(A \times D^X)$	$m \cdot \log n$	Baier, Engelen, Majster-Cederbaum 2000
		=	$m \cdot \log n$ Groote, Verduzco, de Vink 2018
Colour Refinement	$\mathcal{B}X$	$m \cdot \log n$	Alami, Grohe 2017
Weighted Tree Automata	$M(\Sigma X)$	$m \cdot \log^2 m$	Alami, Grohe, Maletti, May 2007
	$M$ non-cancellative		
	$M(\Sigma X)$	$m \cdot \log m$	$m \cdot \log m$ Högberg, Maletti, May 2007
	$M$ cancellative		

More instances:  
further system types  
& equivalences

E.g. Nominal Automata

Generic & Efficient

Articles & Tool  
[tinyurl.com/coalgebra](http://tinyurl.com/coalgebra)

$\Sigma$  fixed



System	Functor $FX$	Run-Time ( $m \geq n$ )		Specific algorithm	
Transition Systems	$\mathcal{P}_f X$	$m \cdot \log n$	$=$	$m \cdot \log n$	Paige, Tarjan 1987
LTS	$\mathcal{P}_f(\mathbb{N} \times X)$	$m \cdot \log m$	$=$ $>$	$m \cdot \log m$ $m \cdot \log n$	Dovier, Piazza, Policriti 2004 Valmari 2009
Markov Chains	$\mathbb{R}^{(X)}$	$m \cdot \log n$	$=$	$m \cdot \log n$	Valmari, Franceschinis 2010
DFA	$2 \times X^A$ ( $A$ fixed)	$n \cdot \log n$	$=$	$n \cdot \log n$	Hopcroft 1971
	$2 \times \mathcal{P}_f(A \times X)$	$ A  \cdot n \cdot \log n$	$=$	$ A  \cdot n \cdot \log n$	Gries 1973/Knuutila 2001
Segala Systems	$\mathcal{P}_f(A \times \mathcal{D}X)$	$m_{\mathcal{D}} \cdot \log m_{\mathcal{P}_f}$	$<$ $=$	$m \cdot \log n$ $m_{\mathcal{D}} \cdot \log m_{\mathcal{P}_f}$	Baier, Engelen, Majster-Cederbaum 2000 Groote, Verduzco, de Vink 2018
Colour Refinement	$\mathcal{B}X$	$m \cdot \log n$	$=$	$m \cdot \log n$	Berkholz, Bonsma, Grohe 2017
Weighted Tree Automata	$M^{(\Sigma X)}$ $M$ non-cancellative	$m \cdot \log^2 m$	$<$	$m \cdot n$	Högberg, Maletti, May 2007
	$M^{(\Sigma X)}$ $M$ cancellative	$m \cdot \log m$	$=$ $\Sigma$ fixed	$m \cdot \log n$	Högberg, Maletti, May 2007

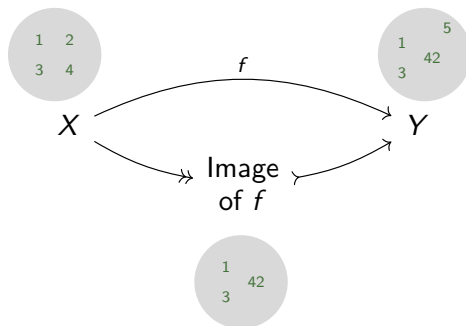
Appendix ...

# Factorizations

$x_1, x_2$  in the same block  $\iff f(x_1) = f(x_2)$

## Kernel pairs

$$\ker f = \{(x_1, x_2) \in X^2 \mid f(x_1) = f(x_2)\}$$

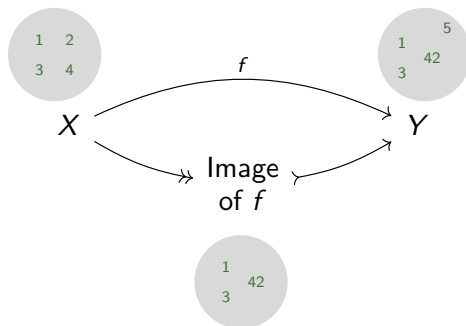


# Factorizations

$x_1, x_2$  in the same block  $\iff f(x_1) = f(x_2)$

## Kernel pairs

$$\ker f = \{(x_1, x_2) \in X^2 \mid f(x_1) = f(x_2)\}$$

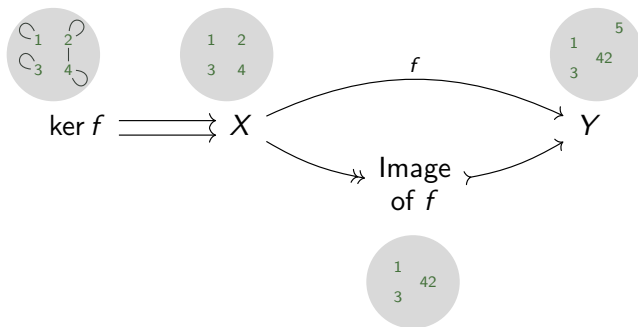


# Factorizations

$x_1, x_2$  in the same block  $:\iff f(x_1) = f(x_2)$

## Kernel pairs

$$\ker f = \{(x_1, x_2) \in X^2 \mid f(x_1) = f(x_2)\}$$

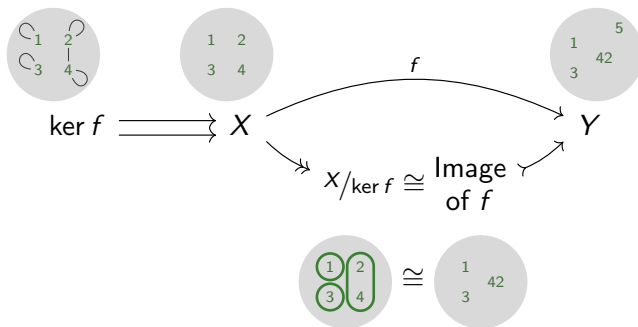


# Factorizations

$x_1, x_2$  in the same block  $\iff f(x_1) = f(x_2)$

## Kernel pairs

$$\ker f = \{(x_1, x_2) \in X^2 \mid f(x_1) = f(x_2)\}$$



### Algorithm for a finite $c: C \rightarrow FC$

- $C/Q := \{C\}$
- $P := \ker(C \xrightarrow{c} FC \xrightarrow{F!} F1)$
- While  $P$  properly finer than  $Q$ :
  - Pick  $S \subsetneq B$ ,  $S \in C/P$ ,  $B \in C/Q$ ,  $|S| \leq \frac{1}{2} \cdot |B|$
  - $C/Q := C/Q - \{B\} \cup \{S, B \setminus S\}$
  - $P := P \cap \ker(C \xrightarrow{c} FC \xrightarrow{F\chi_S^B} F3)$
- Return  $C/P$

### Correctness

If  $F$  is zippable, then the above algorithm computes the simple quotient of  $c: C \rightarrow FC$ .

## Functor encoding

- internal weights  $W$ ,  $w : FX \rightarrow \mathcal{P}_f X \rightarrow W$
- edge labels  $L$
- $b : FX \rightarrow \mathcal{B}(L \times X)$
- update :  $\mathcal{B}(L) \times W \longrightarrow W \times F(2 \times 2) \times W$



Functor:	$G(-)$	$\mathcal{B}$	$\mathcal{D}$	$\mathcal{P}_f$	$F_\Sigma$
Labels $L$ :	$G$	$\mathbb{N}$	$[0, 1]$	1	$\mathbb{N}$
Weights $W$ :	$G^{(2)}$	$\mathcal{B}2$	$\mathcal{D}2$	$\mathbb{N}$	$F_\Sigma 2$
$w(C)$ , $C \subseteq Y$ :	$G\chi_C$	$\mathcal{B}\chi_C$	$\mathcal{D}\chi_C$	$ C \cap (-) $	$F_\Sigma \chi_C$



1. Assume  
everything  
equivalent

1. Assume  
everything  
equivalent



2. Have a  
quotient  
on  $C$

1. Assume  
everything  
equivalent

2. Have a  
quotient  
on  $C$

3. Unravel  
 $c : C \rightarrow FC$   
by one step

```
graph LR; A[1. Assume everything equivalent] --> B[2. Have a quotient on C]; B --> C[3. Unravel c : C -> FC by one step]
```

1. Assume everything equivalent

2. Have a quotient on  $C$

3. Unravel  $c : C \rightarrow FC$  by one step

4. Pick some of the new information

refine further

```
graph TD; 1[1. Assume everything equivalent] --> 2[2. Have a quotient on C]; 2 --> 3[3. Unravel c : C -> FC by one step]; 3 --> 4[4. Pick some of the new information]; 4 -- refine further --> 2;
```

1. Assume everything equivalent

$C$   
 $\Downarrow$   
1

2. Have a quotient on  $C$

3. Unravel  $c : C \rightarrow FC$  by one step

4. Pick some of the new information

refine further

```
graph TD; 1[1. Assume everything equivalent] --> 2[2. Have a quotient on C]; 2 --> 3[3. Unravel c : C -> FC by one step]; 3 --> 4[4. Pick some of the new information]; 4 -- refine further --> 2;
```

1. Assume everything equivalent

$C$   
 $\Downarrow$   
 $1$

2. Have a quotient on  $C$

$Q := \ker a$   
 $\Downarrow$   
 $C$   
 $\downarrow^a$   
 $A$

3. Unravel  $c : C \rightarrow FC$  by one step

refine further

4. Pick some of the new information

1. Assume everything equivalent

$$C$$

$$\Downarrow$$

$$1$$

2. Have a quotient on  $C$

$$Q := \ker a$$

$$\Downarrow$$

$$C$$

$$\downarrow a$$

$$A$$

3. Unravel  $c : C \rightarrow FC$  by one step

$$P := \ker(Fa \cdot c)$$

$$\Downarrow$$

$$C$$

$$\downarrow c$$

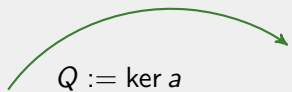
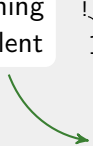
$$FC$$

$$\downarrow Fa$$

$$FA$$

refine further

4. Pick some of the new information



1. Assume everything equivalent

$C$   
 $\Downarrow$   
 $1$

2. Have a quotient on  $C$

$Q := \ker a$   
 $\Downarrow$   
 $C$   
 $\downarrow a$   
 $A$

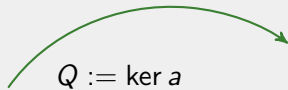
3. Unravel  $c : C \rightarrow FC$  by one step

$P := \ker(Fa \cdot c)$   
 $\Downarrow$   
 $C$   
 $\downarrow c$   
 $FC$   
 $\downarrow Fa$   
 $FA$

refine further

4. Pick some of the new information

$C$   
 $\downarrow$   
 $\Downarrow$   
 $C/P$   
 $\downarrow$   
 $\Downarrow$   
 $C/Q$





1. Assume everything equivalent

$C$   
 $\Downarrow$   
 $1$

2. Have a quotient on  $C$

$Q := \ker a$   
 $\Downarrow$   
 $C$   
 $\downarrow a$   
 $A$

3. Unravel  $c : C \rightarrow FC$  by one step

$P := \ker(Fa \cdot c)$   
 $\Downarrow$   
 $C$   
 $\downarrow c$   
 $FC$   
 $\downarrow Fa$   
 $FA$

refine further

4. Pick some of the new information

$C$   
 $\downarrow$   
 $C/P \rightarrow B$   
 $\downarrow$   
 $C/Q$

$\xrightarrow{b}$  (dashed red arrow from  $C$  to  $B$ )

$\xrightarrow{\text{heuristic}}$  (curved arrow from  $C/Q$  to  $B$ )

1. Assume everything equivalent

$$C \begin{array}{l} \Downarrow \\ \Downarrow \\ 1 \end{array}$$

2. Have a quotient on  $C$

$$Q := \ker a \begin{array}{l} \Downarrow \\ \Downarrow \\ C \\ \downarrow a \\ A \end{array}$$

3. Unravel  $c : C \rightarrow FC$  by one step

$$P := \ker(Fa \cdot c) \begin{array}{l} \Downarrow \\ \Downarrow \\ C \\ \downarrow c \\ FC \\ \downarrow Fa \\ FA \end{array}$$

$a' = \langle a, b \rangle$

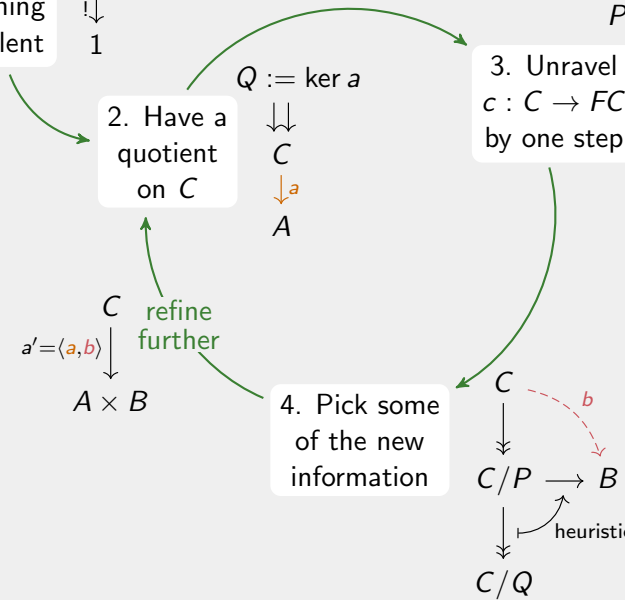
$$C \begin{array}{l} \downarrow \\ A \times B \end{array}$$

refine further

4. Pick some of the new information

$$C \begin{array}{l} \downarrow \\ C/P \end{array} \xrightarrow{b} B \begin{array}{l} \downarrow \\ C/Q \end{array}$$

heuristic



1. Assume everything equivalent

$$C \Downarrow 1$$

2. Have a quotient on  $C$

$$Q := \ker a \\ C \Downarrow A \\ \downarrow a$$

3. Unravel  $c : C \rightarrow FC$  by one step

$$P := \ker(Fa \cdot c) \\ C \Downarrow FC \xrightarrow{c} \\ \downarrow Fa \\ FA$$

$$C \xrightarrow{a' = \langle a, b \rangle} A \times B$$

refine further

4. Pick some of the new information

$$C \xrightarrow{b} C/P \xrightarrow{\text{heuristic}} C/Q \rightarrow B$$

id on  $C/P$ :  
use all new information

use smaller half

# Genericity: Initial partiton

Given

$$C \xrightarrow{c} FC$$

Usual partition refinement algorithms

Return coarsest partition compatible with  $c$ , refining  $C \xrightarrow{\kappa} \mathcal{I}$

# Genericity: Initial partiton

Given

$$C \xrightarrow{c} FC$$

Usual partition refinement algorithms

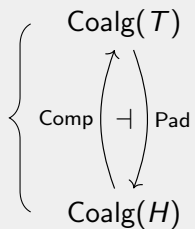
Return coarsest partition compatible with  $c$ , refining  $C \xrightarrow{\kappa} \mathcal{I}$



Coalgebraic partition refinement for  $\mathcal{I} \times F$

For the coalgebra  $C \xrightarrow{\langle \kappa, c \rangle} \mathcal{I} \times FC$

Schröder, Pattinson 2011



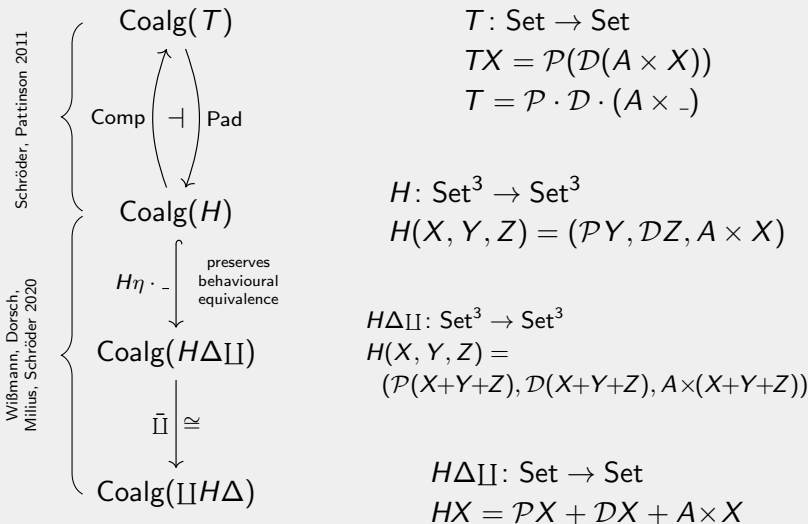
$$T: \text{Set} \rightarrow \text{Set}$$

$$TX = \mathcal{P}(\mathcal{D}(A \times X))$$

$$T = \mathcal{P} \cdot \mathcal{D} \cdot (A \times -)$$

$$H: \text{Set}^3 \rightarrow \text{Set}^3$$

$$H(X, Y, Z) = (\mathcal{P}Y, \mathcal{D}Z, A \times X)$$



## Assumptions

$H$  mono-preserving,  $\mathcal{C}$  extensive, (RegEpi, Mono)-factorization  
 $\Pi: \mathcal{C}^n \rightarrow \mathcal{C} \quad \vdash \quad \Delta: \mathcal{C} \rightarrow \mathcal{C}^n \quad \eta: \text{Id}_{\mathcal{C}^n} \hookrightarrow \Delta\Pi$

## Genericity: Composition

If  $F$  finitary,

$$C \xrightarrow{c} FG C$$



# Genericity: Composition

If  $F$  finitary,

$$C \xrightarrow{c} FG C \quad \rightsquigarrow \quad D \xrightarrow{d} GC$$

# Genericity: Composition

If  $F$  finitary,

$$\begin{array}{ccc} C & \xrightarrow{c} & FG C \\ & \searrow^{c'} & \uparrow Fd \\ & & FD \end{array} \quad \rightsquigarrow \quad D \xrightarrow{d} GC$$

# Genericity: Composition

If  $F$  finitary,

$$\begin{array}{ccc}
 C & \xrightarrow{c} & FG C \\
 & \searrow^{c'} & \uparrow Fd \\
 & & FD
 \end{array}
 \quad \rightsquigarrow \quad
 D \xrightarrow{d} GC$$

A coalgebra on  $\text{Set}^2$  for the functor  $(X, Y) \mapsto (FY, GX)$ :

$$(C, D) \xrightarrow{(c', d)} (FD, GC)$$

# Genericity: Composition

If  $F$  finitary,

$$\begin{array}{ccc}
 C & \xrightarrow{c} & FG C \\
 & \searrow^{c'} & \uparrow Fd \\
 & & FD
 \end{array}
 \quad \rightsquigarrow \quad
 D \xrightarrow{d} GC$$

A coalgebra on  $\text{Set}^2$  for the functor  $(X, Y) \mapsto (FY, GX)$ :

$$(C, D) \xrightarrow{(c', d)} (FD, GC)$$

## Examples

$$\begin{array}{ll}
 \mathcal{P}_f \cdot (A \times (-)) & (2 \times \mathcal{P}_f) \cdot (A \times (-)) \\
 \mathcal{P}_f \cdot (A \times (-)) \cdot \mathcal{D} & \mathcal{P}_f \cdot \mathcal{D} \cdot (A \times (-)) \quad \dots
 \end{array}$$

$$A \xleftarrow{a} X \xrightarrow{b} B$$

$\ker a \cup \ker b$  a kernel in Set

$\Leftrightarrow \ker a \cup \ker b$  transitive

$\Leftrightarrow \forall x \in X : [x]_a \subseteq [x]_b$  or  $[x]_a \supseteq [x]_b$

Example



Non-Example



$$A \xleftarrow{a} X \xrightarrow{b} B$$

$\ker a \cup \ker b$  a kernel in Set

$\Leftrightarrow \ker a \cup \ker b$  transitive

$\Leftrightarrow \forall x \in X : [x]_a \subseteq [x]_b$  or  $[x]_a \supseteq [x]_b$

Example



Non-Example

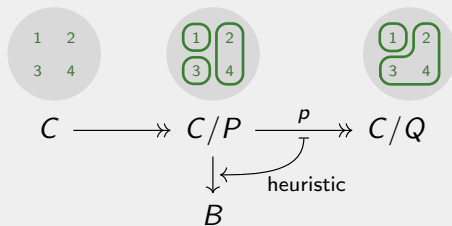


Process smaller half for  $X \xrightarrow{f} F \xrightarrow{g} G$

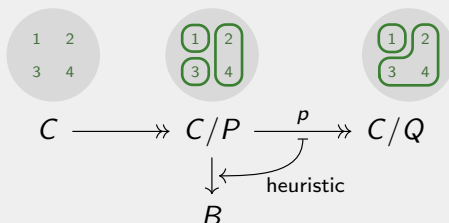
Find  $x \in X$ , with  $S := [x]_f$ ,  $C := [x]_{gf}$ , such that  $2 \cdot |S| \leq |C|$ .

Return  $\langle \chi_S, \chi_C \rangle : X \rightarrow 2 \times 2$

# Heuristic



# Heuristic



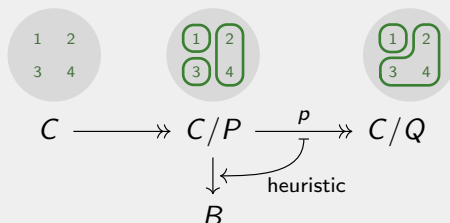
Use all new information

$B = C/P \rightsquigarrow$  Final Chain algorithm

König, Küpper '14



# Heuristic



Use all new information

$B = C/P \rightsquigarrow$  Final Chain algorithm

König, Küpper '14

Process the smaller half

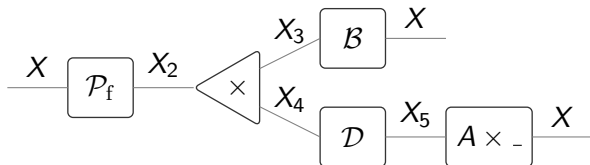
Surrounding block in  $C/Q$

Let  $S \in C/P$ , such that  $2 \cdot |S| \leq |p(S)|$

$B = \{ \overset{\{3\}}{\text{ChosenBlock}}, \overset{\{2, 4\}}{\text{SameSurroundingBlock}}, \overset{\{1\}}{\text{RemainingBlocks}} \}$

# Modularity – for more complicated compositions

$$FX = \mathcal{P}_f(\mathcal{B}X \times \mathcal{D}(A \times X))$$



$$\Rightarrow H: \text{Set}^5 \rightarrow \text{Set}^5$$

$$H(X, X_2, X_3, X_4, X_5) = (\mathcal{P}_f X_2, X_3 \times X_4, \mathcal{B}X, \mathcal{D}X_5, A \times X)$$

$$\Rightarrow H': \text{Set} \rightarrow \text{Set}$$

$$H'X = \mathcal{P}_f X + X \times X + \mathcal{B}X, \mathcal{D}X + A \times X$$

## References

- [BBG17] Christoph Berkholtz, Paul S. Bonsma, Martin Grohe. “Tight Lower and Upper Bounds for the Complexity of Canonical Colour Refinement”. In: **Theory Comput. Syst.** 60.4 (2017), pp. 581–614. DOI: [10.1007/s00224-016-9686-0](https://doi.org/10.1007/s00224-016-9686-0). URL: <https://doi.org/10.1007/s00224-016-9686-0>.
- [BEM00] Christel Baier, Bettina Engelen, Mila Majster-Cederbaum. “Deciding Bisimilarity and Similarity for Probabilistic Processes”. In: **J. Comput. Syst. Sci.** 60 (2000), pp. 187–231.

- [DMSW19] Hans-Peter Deifel, Stefan Milius, Lutz Schröder, Thorsten Wißmann. “Generic Partition Refinement and Weighted Tree Automata”. In: **Formal Methods – The Next 30 Years**. Ed. by Maurice H. ter Beek, Annabelle McIver, José N. Oliveira. Cham: Springer International Publishing, Oct. 2019, pp. 280–297. ISBN: 978-3-030-30942-8. DOI: 10.1007/978-3-030-30942-8\_18. URL: <https://arxiv.org/abs/1811.08850>.
- [DPP04] Agostino Dovier, Carla Piazza, Alberto Policriti. “An efficient algorithm for computing bisimulation equivalence”. In: **Theor. Comput. Sci.** 311.1-3 (2004), pp. 221–256.
- [Gri73] David Gries. “Describing an algorithm by Hopcroft”. In: **Acta Inf.** 2 (1973), pp. 97–109. ISSN: 1432-0525.

- [GVdV18] Jan Groote, Jao Verduzco, Erik de Vink. “An Efficient Algorithm to Determine Probabilistic Bisimulation”. In: **Algorithms** 11.9 (2018), p. 131. DOI: 10.3390/a11090131. URL: <https://doi.org/10.3390/a11090131>.
- [HMM07] Johanna Högberg, Andreas Maletti, Jonathan May. “Bisimulation Minimisation for Weighted Tree Automata”. In: **Proceedings of the 11th International Conference on Developments in Language Theory**. DLT’07. Turku, Finland: Springer-Verlag, 2007, pp. 229–241. ISBN: 978-3-540-73207-5. URL: <http://dl.acm.org/citation.cfm?id=1770310.1770335>.
- [Hop71] John Hopcroft. “An  $n \log n$  algorithm for minimizing states in a finite automaton”. In: **Theory of Machines and Computations**. Academic Press, 1971, pp. 189–196.

- [KK14] Barbara König, Sebastian Küpper. “Generic Partition Refinement Algorithms for Coalgebras and an Instantiation to Weighted Automata”. In: **Theoretical Computer Science, IFIP TCS 2014**. Vol. 8705. LNCS. Springer, 2014, pp. 311–325. ISBN: 978-3-662-44601-0.
- [Knu01] Timo Knuutila. “Re-describing an algorithm by Hopcroft”. In: **Theor. Comput. Sci.** 250 (2001), pp. 333–363. ISSN: 0304-3975.
- [PT87] Robert Paige, Robert Tarjan. “Three partition refinement algorithms”. In: **SIAM J. Comput.** 16.6 (1987), pp. 973–989.
- [SP11] Lutz Schröder, Dirk Pattinson. “Modular Algorithms for Heterogeneous Modal Logics via Multi-Sorted Coalgebra”. In: **Math. Struct. Comput. Sci.** 21.2 (2011), pp. 235–266.

- [Val09] Antti Valmari. “Bisimilarity Minimization in  $\mathcal{O}(m \log n)$  Time”. In: **Applications and Theory of Petri Nets, PETRI NETS 2009**. Vol. 5606. LNCS. Springer, 2009, pp. 123–142. ISBN: 978-3-642-02423-8.
- [VF10] Antti Valmari, Giuliana Franceschinis. “Simple  $\mathcal{O}(m \log n)$  Time Markov Chain Lumping”. In: **Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2010**. Vol. 6015. LNCS. Springer, 2010, pp. 38–52.
- [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>.