

Efficient and Modular Coalgebraic Partition Refinement

Thorsten Wißmann

University Erlangen-Nürnberg

Joint work with:

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

- Published in Concur 2017
- Submitted To LMCS
- Ongoing Implementation

Highlights of Logic, Games and Automata

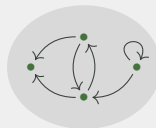
September 19, 2018

Efficient and Modular Coalgebraic Partition Refinement

Efficient and Modular Coalgebraic Partition Refinement

Coalgebras:

State based
systems



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

Efficient and Modular Coalgebraic Partition Refinement

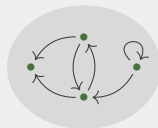
Modularity:

Combine
system
types by

\circ , \times , $+$

Coalgebras:

State based
systems



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

Efficient and Modular Coalgebraic Partition Refinement

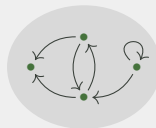
Modularity:

Combine
system
types by

\circ , \times , $+$

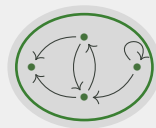
Coalgebras:

State based
systems



Partition Refinement:

Successively distinguish
different behaviour



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

Efficient and Modular Coalgebraic Partition Refinement

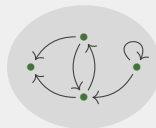
Modularity:

Combine
system
types by

$\circ, \times, +$

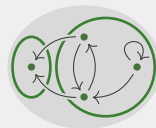
Coalgebras:

State based
systems



Partition Refinement:

Successively distinguish
different behaviour



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

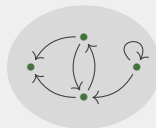
Efficient and Modular Coalgebraic Partition Refinement

Modularity:

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

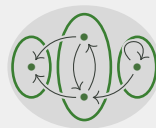
Coalgebras:

State based
systems



Partition Refinement:

Successively distinguish
different behaviour



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

Efficient and Modular Coalgebraic Partition Refinement

Efficiency:

- (a) Incrementally compute partitions
- (b) Complexity Analysis

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

Edges

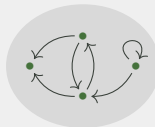
States

Modularity:

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

Coalgebras:

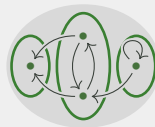
State based systems



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

Partition Refinement:

Successively distinguish different behaviour





Share Common
Structure & Ideas

Similar
Run-Time

Variations in
Details

Share Common Structure & Ideas

Deterministic
Finite Automata

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

Hopcroft '71 Gries '73

Knuutila '01

Similar Run-Time

Variations in Details

Share Common Structure & Ideas

Deterministic
Finite Automata

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

Similar Run-Time

Variations in Details

Transition Systems

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

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

Transition Systems

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

Segala Systems

$m \cdot n \cdot (\log m + \log n)$
Baier, Engelen,
Majster-Cederbaum '00

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

Transition Systems

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

Segala Systems

$m \cdot n \cdot (\log m + \log n)$
Baier, Engelen,
Majster-Cederbaum '00

Labelled
Transition Systems

$m \cdot \log n$
Valmari '09

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

Transition Systems

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

Segala Systems

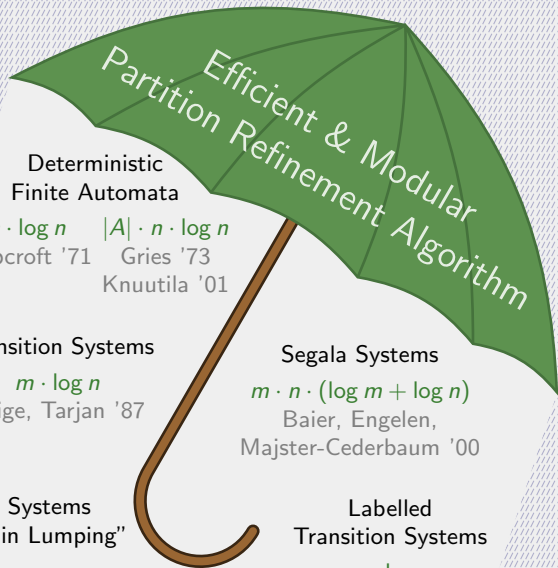
$m \cdot n \cdot (\log m + \log n)$
Baier, Engelen,
Majster-Cederbaum '00

Weighted Systems
"Markov Chain Lumping"

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

Labelled
Transition Systems

$m \cdot \log n$
Valmari '09



Deterministic
Finite Automata

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

Transition Systems

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

Segala Systems

$m \cdot n \cdot (\log m + \log n)$
Baier, Engelen,
Majster-Cederbaum '00

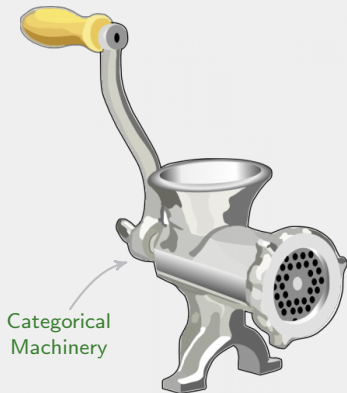
Weighted Systems
"Markov Chain Lumping"

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

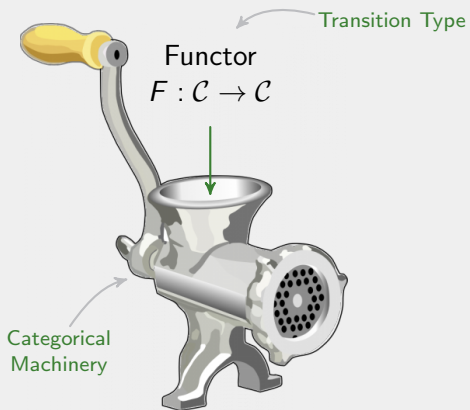
Labelled
Transition Systems

$m \cdot \log n$
Valmari '09

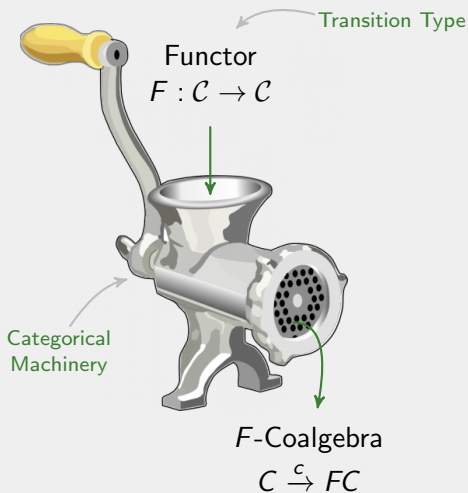
Coalgebra – Generic state based systems



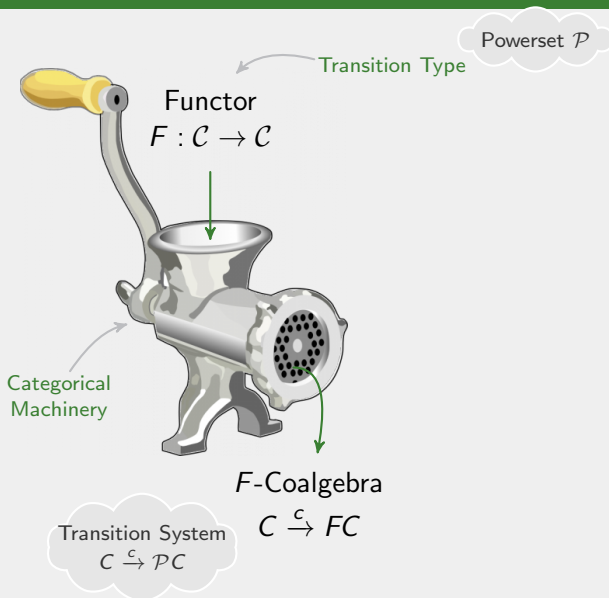
Coalgebra – Generic state based systems



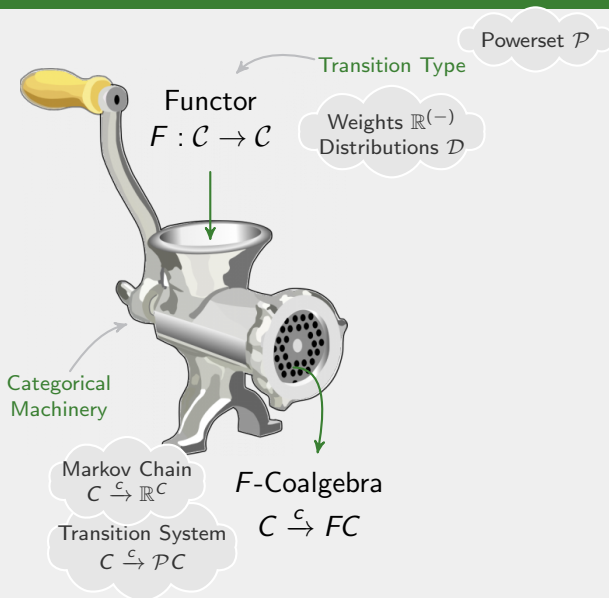
Coalgebra – Generic state based systems



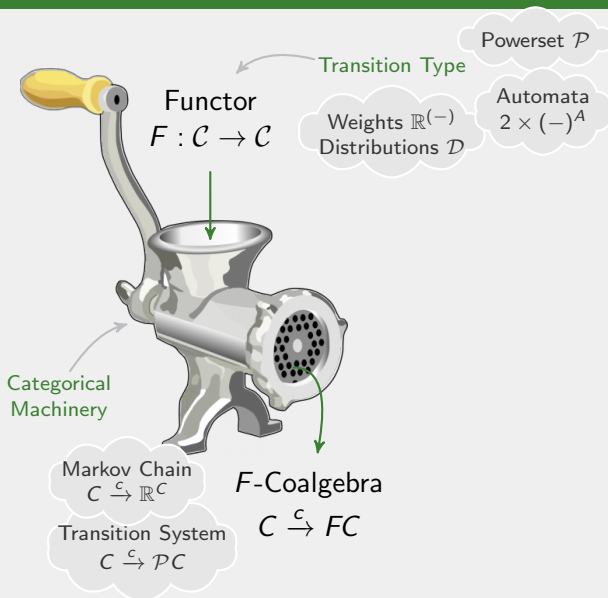
Coalgebra – Generic state based systems



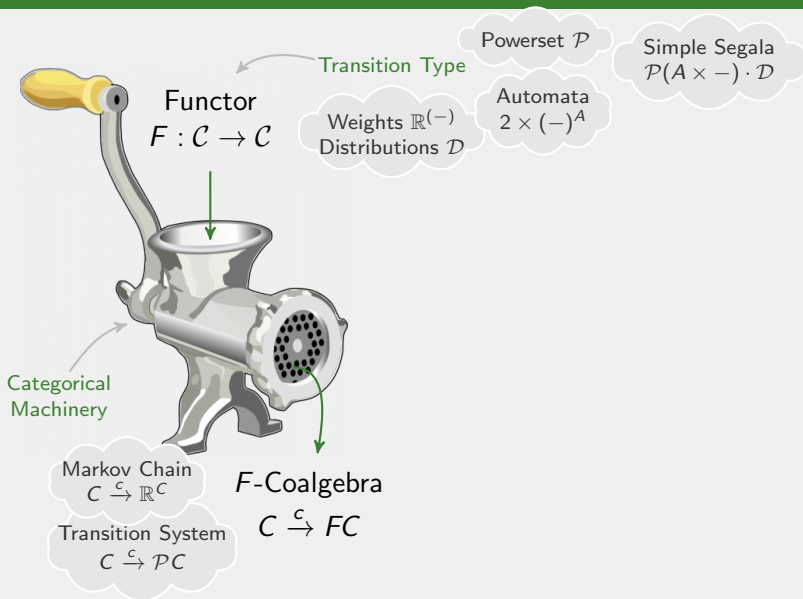
Coalgebra – Generic state based systems



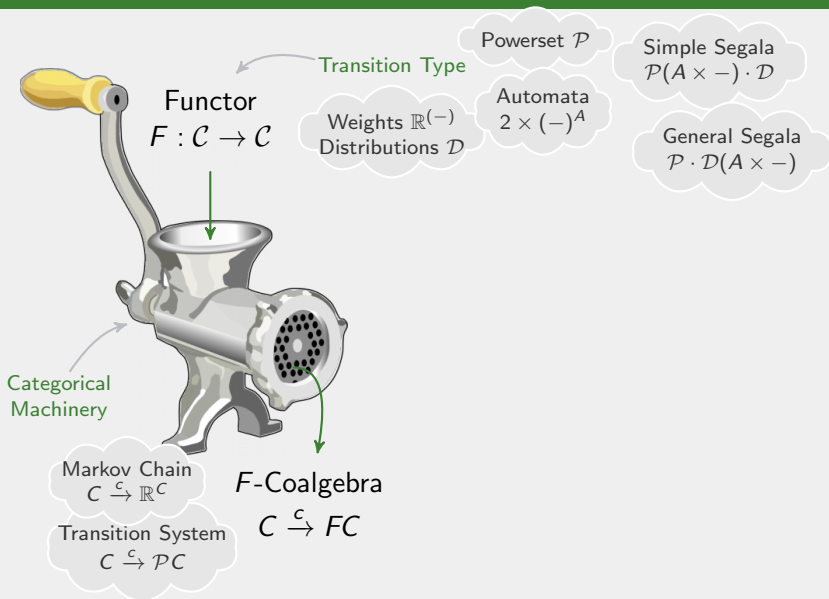
Coalgebra – Generic state based systems



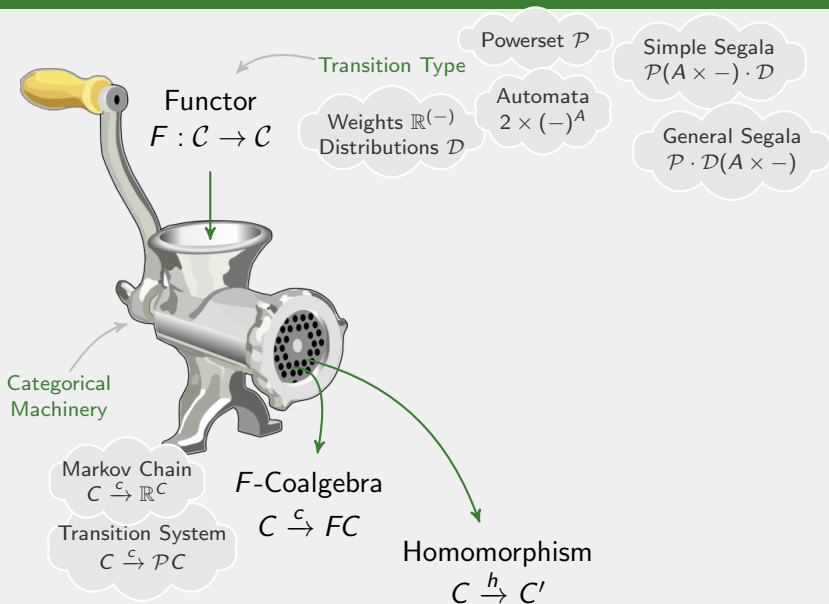
Coalgebra – Generic state based systems



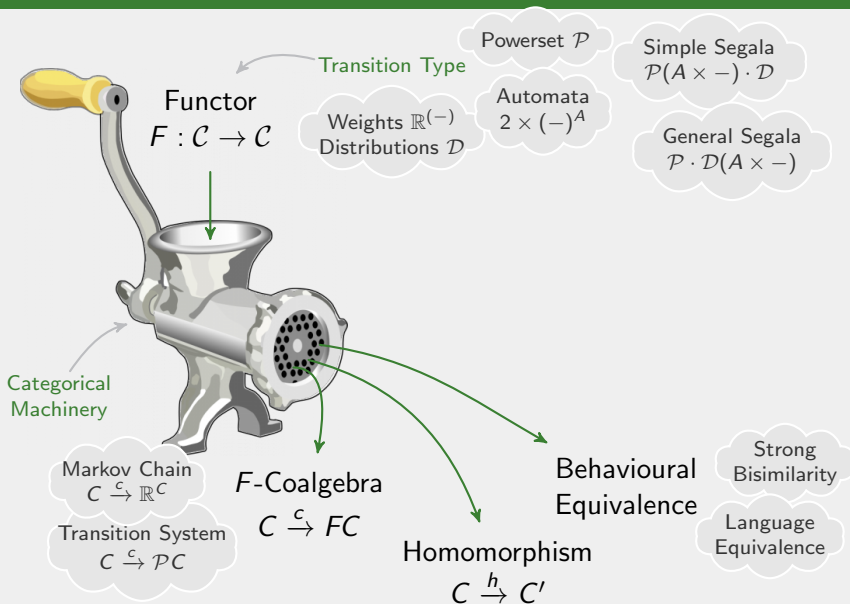
Coalgebra – Generic state based systems



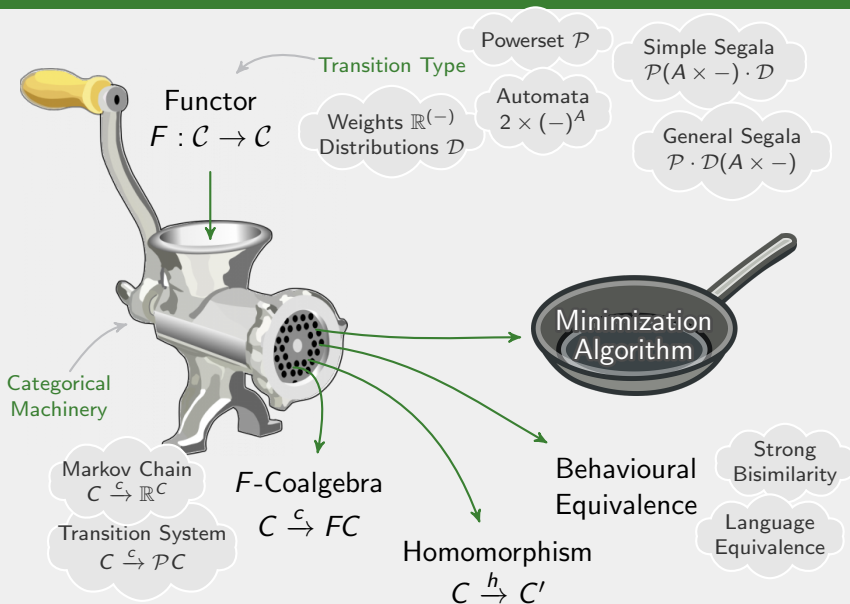
Coalgebra – Generic state based systems



Coalgebra – Generic state based systems



Coalgebra – Generic state based systems



Construction on a category \mathcal{C}



Construction on a category \mathcal{C}



Efficiency

$F: \text{Set} \rightarrow \text{Set}$ is zippable
&
 F has a refinement interface

Pseudo-Code



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

Edges

States

Construction on a category \mathcal{C}



Efficiency

$F: \text{Set} \rightarrow \text{Set}$ is zippable
&
 F has a refinement interface

Pseudo-Code



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

Edges

States

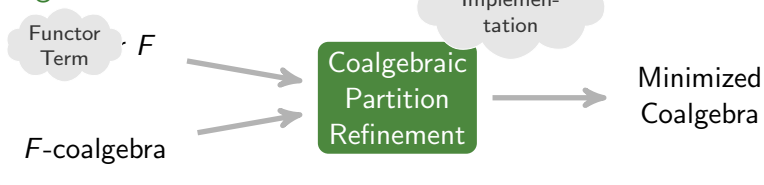
Modularity

$F, H ::= F \cdot H \mid F \times H \mid F + H \mid C \mid \mathcal{P} \mid \mathcal{B}_f \mid G^{(-)}$

Constant Set

Group

Algorithm on Sets



Efficiency

$F: \text{Set} \rightarrow \text{Set}$ is zippable
&
 F has a refinement interface

Pseudo-Code



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

Edges

States

Modularity

Constant Set

Group

$$F, H ::= F \cdot H \mid F \times H \mid F + H \mid C \mid \mathcal{P} \mid \mathcal{B}_f \mid G^{(-)}$$

System	Functor	Concrete algorithm		Our instantiation
Transition Systems	\mathcal{P}	$(m + n) \cdot \log n$ Paige, Tarjan '87	=	$(m + n) \cdot \log n$
LTS	$\mathcal{P}(A \times -)$	$(m + n) \cdot \log(m + n)$ Dovier, Piazza, Policriti '04	=	$(m + n) \cdot \log(m + n)$
		$(m + n) \cdot \log m$ Valmari '09	<	
Markov Chains	$\mathbb{R}(-)$	$(m + n) \cdot \log n$ Valmari, Franceschinis '10	=	$(m + n) \cdot \log n$
DFA	$2 \times (-)^A$	$n \cdot \log n$ for fixed A , Hopcroft '71	=	$n \cdot \log n$
	$2 \times \mathcal{P}(A \times -)$	$ A \cdot n \cdot \log n$, Gries '73, Knuutila '01	\approx	$ A \cdot n \cdot \log n$ $+ A \cdot n \cdot \log A $
Segala Systems	$\mathcal{P}(A \times -) \cdot \mathcal{D}$	$m_{\mathcal{P}} \cdot n \cdot \log(m_{\mathcal{P}} \cdot n)$ Baier, Engelen, Majster-Cederbaum '00	\geq	$(m_{\mathcal{P}} + m_{\mathcal{D}} + n)$ $\cdot \log(m_{\mathcal{P}} + n)$

System	Functor	Concrete algorithm	Our instantiation
Transition Systems	\mathcal{P}	$(m + n) \cdot \log n$ Paige, Tarjan '87	$(m + n) \cdot \log n$
LTS	$\mathcal{P}(A \times -)$	$(m + n) \cdot \log(m + n)$ Dovier, Piazza, Proietti '04 $(m + n) \cdot \log n$ Maimari '02	$(m + n) \cdot \log(m + n)$
Markov Chains	$\mathbb{R}(-)$	$(m + n) \cdot \log n$ Valmari, Franceschini '10	$(m + n) \cdot \log n$
DFA	$2 \times (-)$	$n \cdot \log n$ for fixed A , Hopcroft '71	$n \cdot \log n$
	$2 \times \mathcal{P}(A \times -)$	$ A \cdot n \cdot \log n$, Gries '73, Knuutila '01	$ A \cdot n \cdot \log n$ $+ A \cdot n \cdot \log A $
Segala Systems	$\mathcal{P}(A \times -) \cdot \mathcal{D}$	$m_{\mathcal{P}} \cdot n \cdot \log(m_{\mathcal{P}} \cdot n)$ Baier, Engelen, Majster-Cederbaum '00	$(m_{\mathcal{P}} + m_{\mathcal{D}} + n)$ $\cdot \log(m_{\mathcal{P}} + n)$

Generic & Efficient

System	Functor	Algorithm	Our instantiation
Transition Systems	\mathcal{P}	$(m+n) \cdot \log n$	$(m+n) \cdot \log n$
LTS	$\mathcal{P}(A \times -)$	Dovier, Piazza, Proc. Criti '04 $(m+n) \cdot \log n$ Mimari '02 $(m+n) \cdot \log(m+n)$	$(m+n) \cdot \log(m+n)$
Markov Chains	$\mathbb{R}(-)$	Valmari, Franceschini '10 $(m+n) \cdot \log n$	$(m+n) \cdot \log n$
DFA	$2 \times (-)$	$n \cdot \log n$ for fixed A , Hopcroft '71	$n \cdot \log n$
	$2 \times \mathcal{P}(A \times -)$	$ A \cdot n \cdot \log n$, Gries '73, Knuutila '01	\approx $ A \cdot n \cdot \log n$ $+ A \cdot n \cdot \log A $
Segala Systems	$\mathcal{P}(A \times -) \cdot \mathcal{D}$	$m_{\mathcal{P}} \cdot n \cdot \log(m_{\mathcal{P}} \cdot n)$ Baier, Engelen, Majster-Cederbaum '00	\geq $(m_{\mathcal{P}} + m_{\mathcal{D}} + n)$ $\cdot \log(m_{\mathcal{P}} + n)$

More instances:
further system types
& categories

Generic & Efficient

System	Functor	Algorithm	Our instantiation
Transition Systems	\mathcal{P}	$(m+n) \cdot \log n$	$(m+n) \cdot \log n$
LTS	$\mathcal{P}(A \times -)$	$(m+n) \cdot \log(m+n)$ Dovier, Piazza, Proc. Criti '04 Mimari '02	$(m+n) \cdot \log(m+n)$
Marr Chains	$\mathcal{P}(A \times -)$	$(m+n) \cdot \log n$ Valmari, Franceschini '10	$(m+n) \cdot \log n$
DFA	$2 \times (-)$	$n \cdot \log n$ for fixed A , Hopcroft '71	$n \cdot \log n$
	$2 \times \mathcal{P}(A \times -)$	$ A \cdot n \cdot \log n$, Gries '73, Knuutila '01	$ A \cdot n \cdot \log n$ $+ A \cdot n \cdot \log A $
Segala Systems	$\mathcal{P}(A \times -) \cdot \mathcal{D}$	$m_{\mathcal{P}} \cdot n \cdot \log(m_{\mathcal{P}} \cdot n)$ Baier, Engelen, Majster-Cederbaum '00	$(m_{\mathcal{P}} + m_{\mathcal{D}} + n)$ $\cdot \log(m_{\mathcal{P}} + n)$

More instances:
further system types
& categories

Compare to
existing concrete
implementations

Generic & Efficient

System	Functor	Complexity	Our instantiation
Transition Systems	\mathcal{P}	$(m+n) \cdot \log n$	$(m+n) \cdot \log n$
LTS	\mathcal{P}	$(m+n) \cdot \log(m+n)$	$(m+n) \cdot \log(m+n)$
Marr Chains	$\mathcal{P}(A \times -)$	$(m+n) \cdot \log n$	$(m+n) \cdot \log n$
DFA	$2 \times (-)$	$n \cdot \log n$ for fixed n	$\log n$
	$2 \times \mathcal{P}(A \times -)$	$ A \cdot n$	$\log n$
Segala Systems	$\mathcal{P}(A \times -) \cdot \mathcal{D}$	$m_{\mathcal{P}} \cdot n \cdot \log(m_{\mathcal{P}} + m_{\mathcal{D}} + n)$	$\cdot \log(m_{\mathcal{P}} + n)$

More instances:
further system types
& categories

Compare to
existing concrete
implementations

Generic & Efficient

See poster and
tinyurl.com/coalgebra

System	Functor	Concrete algorithm		Our instantiation
Transition Systems	\mathcal{P}	$(m+n) \cdot \log n$ Paige, Tarjan '87	=	$(m+n) \cdot \log n$
LTS	$\mathcal{P}(A \times -)$	$(m+n) \cdot \log(m+n)$ Dovier, Piazza, Policriti '04	=	$(m+n) \cdot \log(m+n)$
		$(m+n) \cdot \log m$ Valmari '09	<	
Markov Chains	$\mathbb{R}(-)$	$(m+n) \cdot \log n$ Valmari, Franceschinis '10	=	$(m+n) \cdot \log n$
DFA	$2 \times (-)^A$	$n \cdot \log n$ for fixed A , Hopcroft '71	=	$n \cdot \log n$
	$2 \times \mathcal{P}(A \times -)$	$ A \cdot n \cdot \log n$, Gries '73, Knuutila '01	\approx	$ A \cdot n \cdot \log n$ $+ A \cdot n \cdot \log A $
Segala Systems	$\mathcal{P}(A \times -) \cdot \mathcal{D}$	$m_{\mathcal{P}} \cdot n \cdot \log(m_{\mathcal{P}} \cdot n)$ Baier, Engelen, Majster-Cederbaum '00	\geq	$(m_{\mathcal{P}} + m_{\mathcal{D}} + n)$ $\cdot \log(m_{\mathcal{P}} + n)$

Appendix ...

The Coalgebraic Task

For a functor $F : \mathcal{C} \rightarrow \mathcal{C}$

Given a coalgebra

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

no proper
quotient

find the simple quotient

all equivalent
behaviours
identified

The Coalgebraic Task

For a functor $F : \mathcal{C} \rightarrow \mathcal{C}$

Given a coalgebra

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

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 $F : \mathcal{C} \rightarrow \mathcal{C}$

$$\begin{array}{ccc} \text{Given a coalgebra} & C & \xrightarrow{c} & FC \\ & h \downarrow & & \downarrow Fh \\ & C' & \xrightarrow{c'} & FC' \end{array}$$

no proper
quotient

find the simple quotient

all equivalent
behaviours
identified

Instance

Instance

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

Automata
minimization

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

Bisimilarity
minimization

The Coalgebraic Task

For a functor $F : \mathcal{C} \rightarrow \mathcal{C}$

$$\begin{array}{ccc} \text{Given a coalgebra} & C & \xrightarrow{c} & FC \\ & \downarrow h & & \downarrow Fh \\ & C' & \xrightarrow{c'} & FC' \end{array}$$

no proper
quotient

find the simple quotient

all equivalent
behaviours
identified

Instance

Instance

Instance

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

Automata
minimization

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

Bisimilarity
minimization

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

Markov chain
lumping

The Coalgebraic Task

For a functor $F : \mathcal{C} \rightarrow \mathcal{C}$

Given a coalgebra $C \xrightarrow{c} FC$

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

no proper
quotient

find the simple quotient

all equivalent
behaviours
identified

Instance

Instance

Instance

Instance

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

Automata
minimization

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

Bisimilarity
minimization

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

Markov chain
lumping

...

Functors F zippable, if

$$F(L + R) \xrightarrow{\text{unzip}} F(L + 1) \times F(1 + R) \text{ is monic.}$$

E.g. Id, Constants, \times , $+$, \hookrightarrow , $M^{(-)}$, part. additive

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

$$\begin{array}{c} a_1 \ a_2 \ b_1 \ a_3 \ b_2 \xrightarrow{\text{unzip}} \\ \left(\begin{array}{c} a_1 \ a_2 \ - \ a_3 \ -, \\ - \ - \ b_1 \ - \ b_2 \end{array} \right) \longleftarrow \end{array}$$

$(-)^*$ is zippable

$$\begin{array}{c} \{a_1, a_2, b_1\} \xrightarrow{\text{unzip}} \\ \left(\begin{array}{c} \{a_1, a_2, -\}, \\ \{-, b_1\} \end{array} \right) \longleftarrow \end{array}$$

\mathcal{P} is zippable

Functors F zipplable, if

$$F(L + R) \xrightarrow{\text{unzip}} F(L + 1) \times F(1 + R) \text{ is monic.}$$

E.g. Id, Constants, \times , $+$, \hookrightarrow , $M^{(-)}$, part. additive

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

$$\begin{array}{c} a_1 \ a_2 \ b_1 \ a_3 \ b_2 \\ \text{unzip} \\ \left. \begin{array}{l} (a_1 \ a_2 \ - \ a_3 \ - \\ \ - \ b_1 \ - \ b_2) \end{array} \right\} \end{array}$$

$(-)^*$ is zipplable

$$\begin{array}{c} \{a_1, a_2, b_1\} \\ \text{unzip} \\ \left. \begin{array}{l} (\{a_1, a_2, -\}, \\ \{-, b_1\}) \end{array} \right\} \end{array}$$

\mathcal{P} is zipplable

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

$$\begin{array}{c} \text{unzip} \left. \begin{array}{l} \{\{a_1, b_1\}, \{a_2, b_2\}\} \\ \{\{a_1, b_2\}, \{a_2, b_1\}\} \end{array} \right\} \\ \left. \begin{array}{l} (\{\{a_1, -\}, \{a_2, -\}\}, \\ \{-, b_1\}, \{-, b_2\}) \end{array} \right\} \text{unzip} \end{array}$$

\mathcal{PP} is not zipplable

~~Composition~~

~~Quotients~~

Functor encoding

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



Functor:	$G(-)$	\mathcal{B}_f	\mathcal{D}	\mathcal{P}	F_Σ
Labels L :	G	\mathbb{N}	$[0, 1]$	1	\mathbb{N}
Weights W :	$G^{(2)}$	$\mathcal{B}_f 2$	$\mathcal{D} 2$	\mathbb{N}	$F_\Sigma 2$
$w(C)$, $C \subseteq Y$:	G_{χ_C}	$\mathcal{B}_f \chi_C$	\mathcal{D}_{χ_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]; A --> C;
```


1. Assume everything equivalent

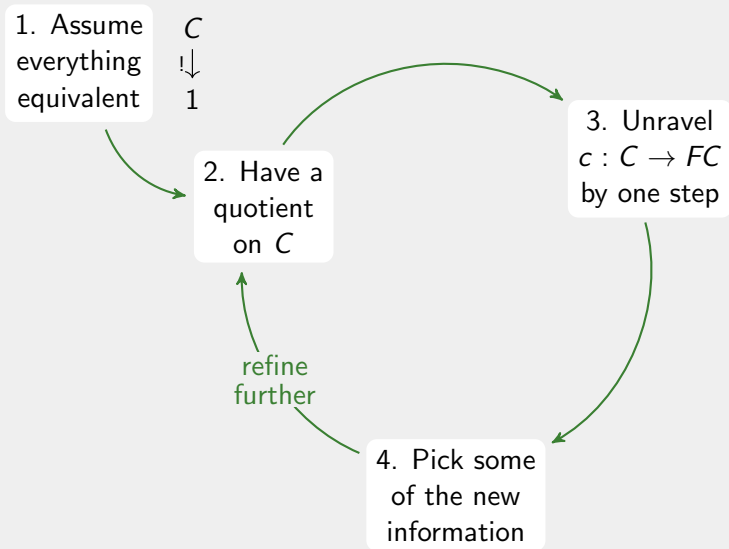
2. Have a quotient on C

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

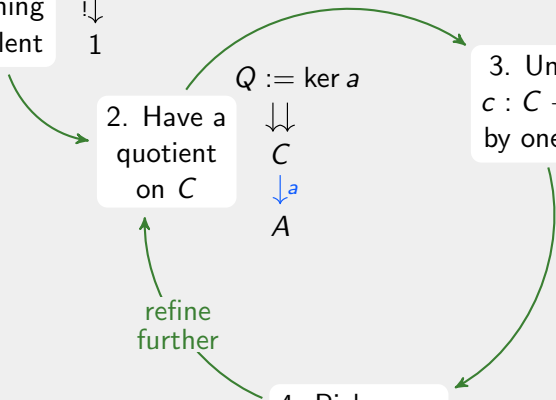
2. Have a quotient on C

$Q := \ker a$
 \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

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

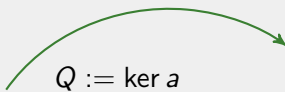
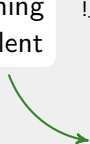
2. Have a quotient on C

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

refine further

4. Pick some of the new information

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

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


1. Assume everything equivalent

$$C \Downarrow 1$$

2. Have a quotient on C

$$Q := \ker a$$

$$C \Downarrow A$$

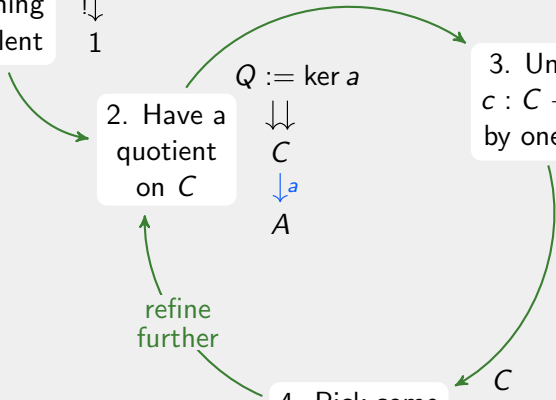
refine further

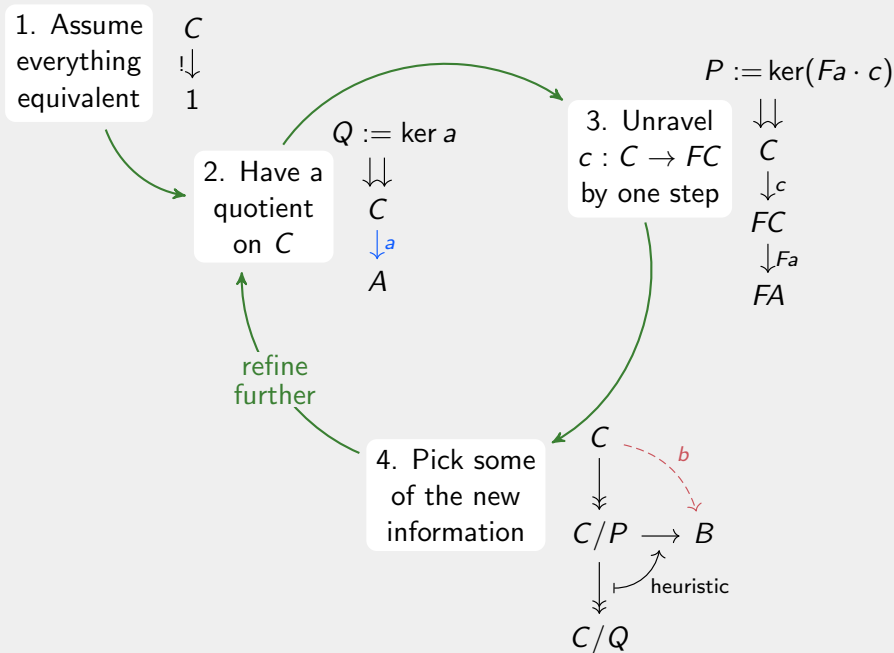
4. Pick some of the new information

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

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

$$C \Downarrow FC \xrightarrow{Fa} FA$$

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




1. Assume everything equivalent

$$C \Downarrow 1$$

2. Have a quotient on C

$$Q := \ker a \Downarrow C \xrightarrow{a} A$$

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

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

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

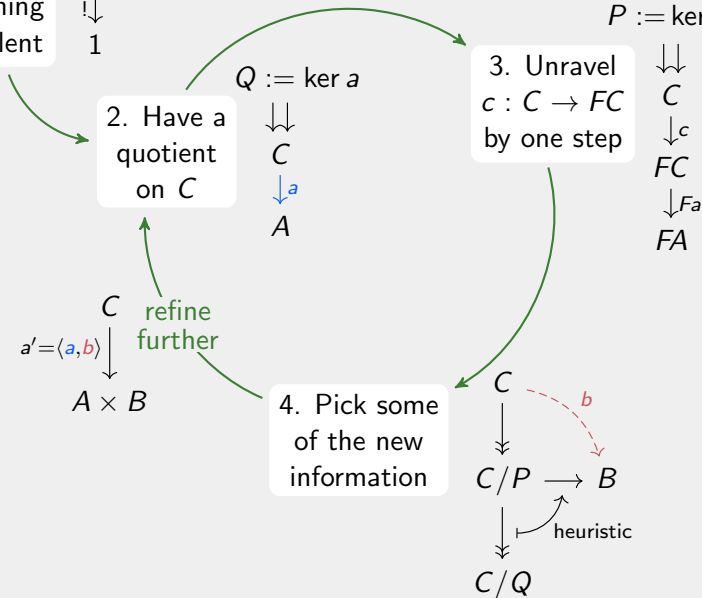
refine further

4. Pick some of the new information

$$C \xrightarrow{c} FC \xrightarrow{Fa} FA$$

$$C \xrightarrow{c} C/P \xrightarrow{b} B$$

$$C/P \xrightarrow{\text{heuristic}} C/Q$$



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

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

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

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

Factorizations

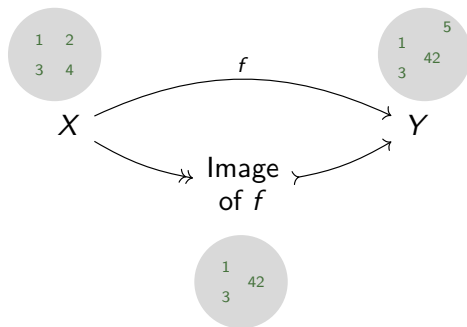
Equivalence Relations	\cong	Quotients \cong Partitions
Kernels	\cong	Regular Epimorphisms

Factorizations

Equivalence Relations \cong Quotients \cong Partitions

Kernels \cong Regular Epimorphisms

Category with (Regular Epi, Mono)-Factorizations

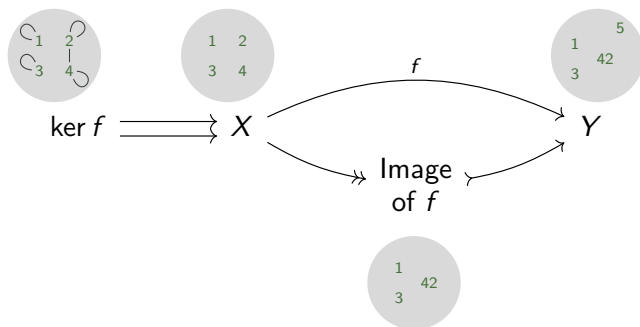


Factorizations

Equivalence Relations \cong Quotients \cong Partitions

Kernels \cong Regular Epimorphisms

Category with (Regular Epi, Mono)-Factorizations



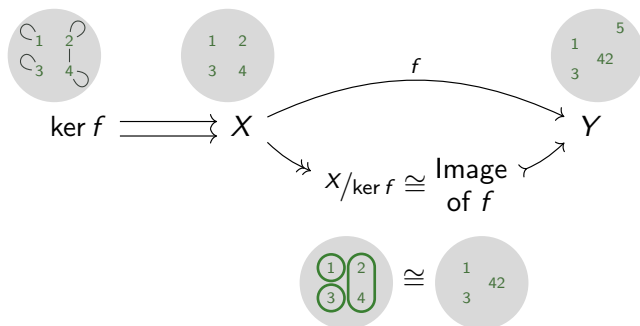
$$\ker f = \{(x_1, x_2) \in X^2 \mid f(x_1) = f(x_2)\}$$

Factorizations

Equivalence Relations \cong Quotients \cong Partitions

Kernels \cong Regular Epimorphisms

Category with (Regular Epi, Mono)-Factorizations



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

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 partition

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

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 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 Set^2 for the functor $(X, Y) \mapsto (FY, GX)$:

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

Examples

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

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

$\ker a \cup \ker b$ 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

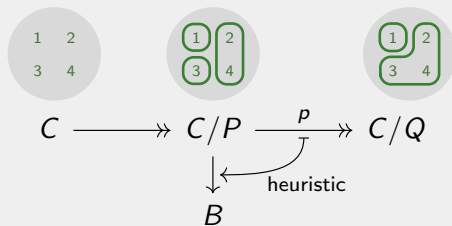


Process smaller half for $X \xrightarrow{f} F \xrightarrow{g} G$

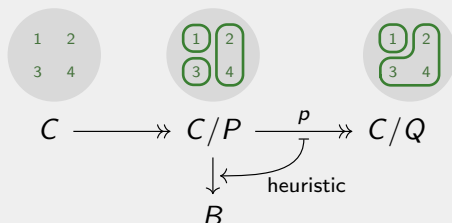
Find $x \in X$, with $S := [x]_f$, $C := [x]_{gf}$, such that $2 \cdot |S| \leq |C|$.

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

Heuristic



Heuristic

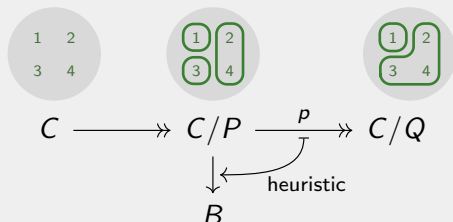


Use all new information

$B = C/P \rightsquigarrow$ 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 $2 \cdot |S| \leq |p(S)|$

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

References

- [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.
- [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.
- [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.
- [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.