# Efficient and Modular Coalgebraic Partition Refinement


T.CS Chair of Theoretical Computer Science

Thorsten Wißmann

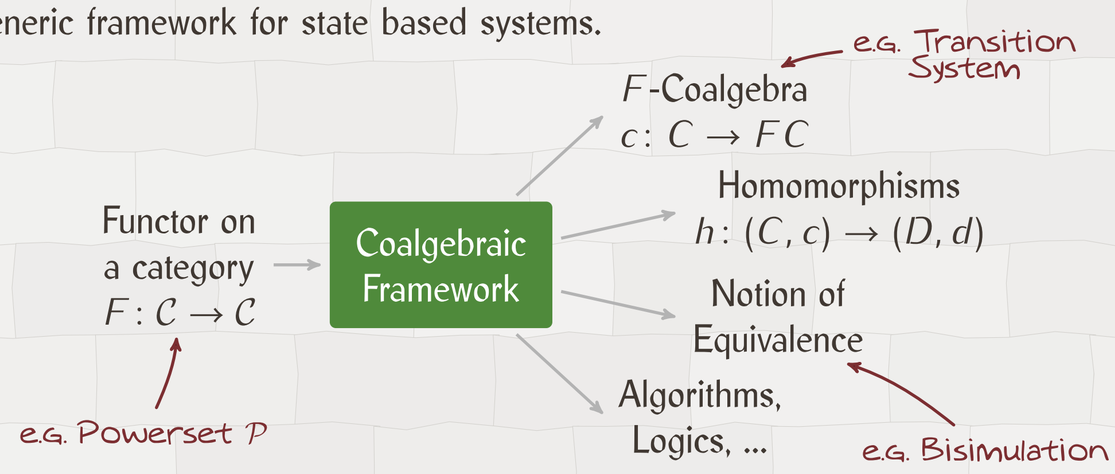
joint work with:

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

FAU Friedrich-Alexander-University Erlangen-Nürnberg


## Coalgebra

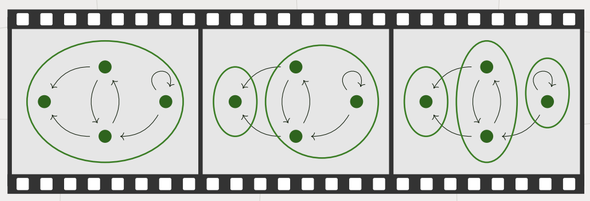
Generic framework for state based systems.




## Partition Refinement Algorithms

State-based System → Partition Refinement → Minimized System

1. Take singleton partition
2. Iterate: refine partition using one step of transition structure



## Generic Coalgebraic Partition Refinement

1. Assume everything equivalent: $C \downarrow !\ 1$

2. Have a quotient on $C$: $Q \rightrightarrows C \xrightarrow{a} A$ (ker(a); actually just some map)

3. Unravel $c : C \to FC$ by one step: $P \rightrightarrows C \xrightarrow{c} FC \xrightarrow{Fa} FA$ (Crucial step of the algorithm; ker(Fa·c))

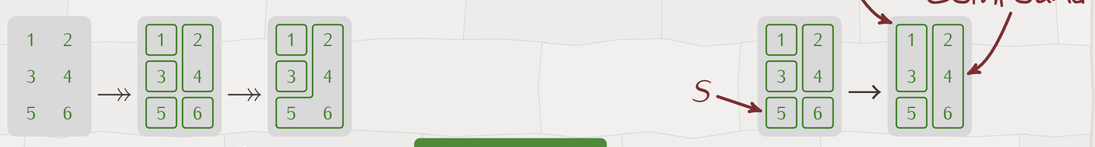
   - if $P = Q$: $Q \rightrightarrows C$ relates all behaviourally equivalent states of $c: C \to FC$
   - if $P$ finer than $Q$:

4. Pick some of the new information

$$C \twoheadrightarrow C/P \twoheadrightarrow C/Q$$

($b$ and $k\downarrow$ to $B$; heuristic)

refine further $C \xrightarrow{\langle a,b \rangle} A \times B$




$$C \twoheadrightarrow C/P \overset{p}{\twoheadrightarrow} C/Q \to \text{Heuristics} \to k: C/P \to B$$

Examples:

- Identity on $C/P$: final chain
- "Smaller half": Choose $S \in C/P$ with $2 \cdot |S| \leq |p(S)|$ (already in Hopcroft '71)

and set $k: C/P \to \{\text{Choice}, \text{SameCompound}, \text{Remaining}\}$

## Efficiency

$F: \text{Set} \to \text{Set}$ zippable & $F$ has refinement-interface $\Longrightarrow$ Algorithm for incremental computation with $\mathcal{O}\big((m+n) \cdot \log n\big)$ run-time

$F(A+B) \to F(A+1) \times F(1+B)$ injective

E.g.: $\mathcal{P}_f$, $\mathcal{D}$, $F_\Sigma$, $\mathbb{R}^{(-)}$, $\mathcal{B}$, ...

Non-Example: $\mathcal{P}_f \cdot \mathcal{P}_f$

Sets $A$ "Labels", $W$ "Weights"
init: $H1 \times \mathcal{B}A \to W$
update: $\mathcal{B}A \times W \to W \times H3 \times W$ (Need to run in linear time)

Hard-wire "Smaller half" heuristics to incrementally compute $P$ fast in step 3.

(Edges; States)

## Instances

| System | Functor | Specific algorithm | | Our instantiation |
|---|---|---|---|---|
| Transition Systems | $\mathcal{P}_f$ | $(m+n) \cdot \log n$ Paige, Tarjan '87 | $=$ | $(m+n) \cdot \log n$ |
| Labelled Transition Systems | $\mathcal{P}_f(A \times -)$ | $(m+n) \cdot \log(m+n)$ 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 | $=$ | $n \cdot \log n$ |
| | $2 \times \mathcal{P}_f(A \times -)$ | $\lvert A \rvert \cdot n \cdot \log n$, Gries '73, Knuutila '01 | $\approx$ | $\lvert A \rvert \cdot n \cdot \log n + \lvert A \rvert \cdot n \cdot \log \lvert A \rvert$ |
| Segala Systems | $\mathcal{P}_f(A \times -) \cdot \mathcal{D}$ | $m_\mathcal{P} \cdot n \cdot \log(m_\mathcal{P} + n)$ Baier, Engelen, Majster-Cederbaum'00 | $\geq$ | $(m_\mathcal{P} + m_\mathcal{D} + n) \cdot \log(m_\mathcal{P} + n)$ |

## Modularity

We can reduce any functor build by (Constant Set; Group)

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

to a zippable functor with a refinement-interface.

## Generic Implementation

Functor Term $F$ / $F$-coalgebra → Haskell Implementation → Minimized Coalgebra

Resources:

- Concur 2017 Conference Paper
  arxiv.org/abs/1705.08362
- Journal Paper Submitted to LMCS
  arxiv.org/abs/1806.05654
- Ongoing Implementation

tinyurl.com/coalgebra