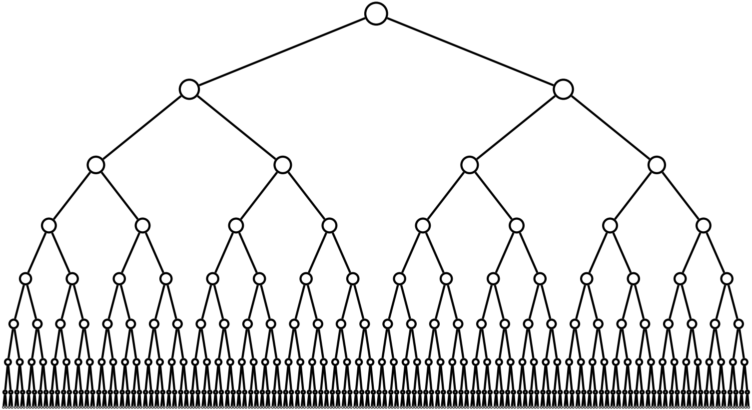
Puzzle: Colouring Infinite Trees

Consider the infinite binary tree:



Puzzle: Colouring Infinite Trees

Corecursion without Bisimulation

Venanzio Capretta

Colouring Infinite Trees

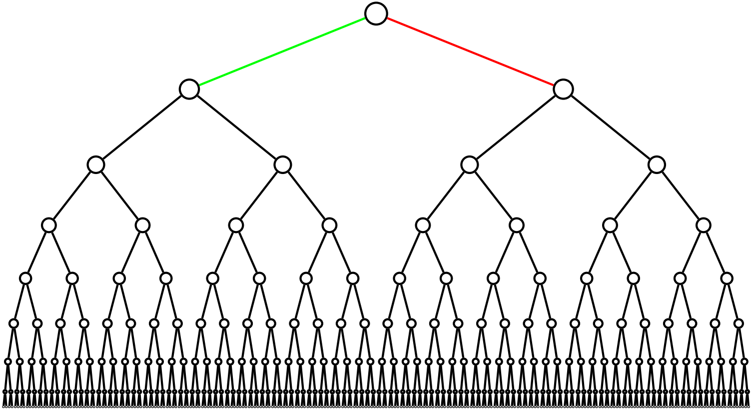
Concepts

pCRA

Bisimulation

Relation between pCRA and DA

Colour the edges green and red.



Puzzle: Colouring Infinite Trees

Corecursion without Bisimulation

Venanzio Capretta

Colouring Infinite Trees

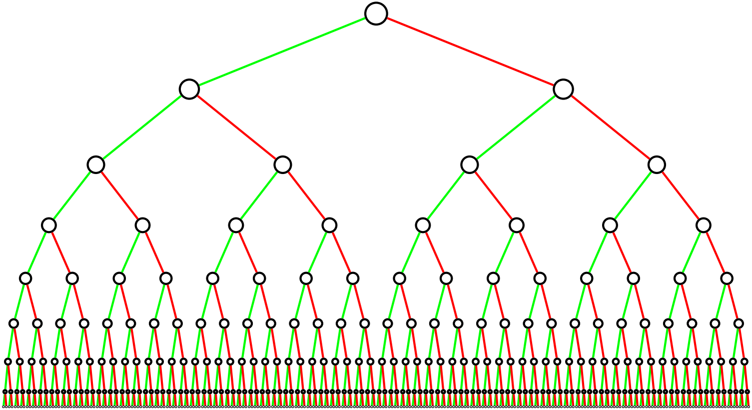
Concepts

pCRA

Bisimulation

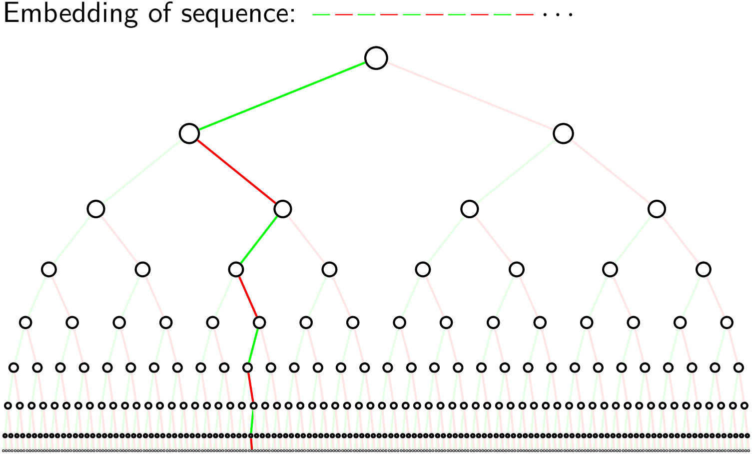
Relation between pCRA and DA

Colour the edges green and red.



Puzzle: Colouring Infinite Trees

Embedding of sequence:



Puzzle: Colouring Infinite Trees

Corecursion without Bisimulation

Venanzio Capretta

Colouring Infinite Trees

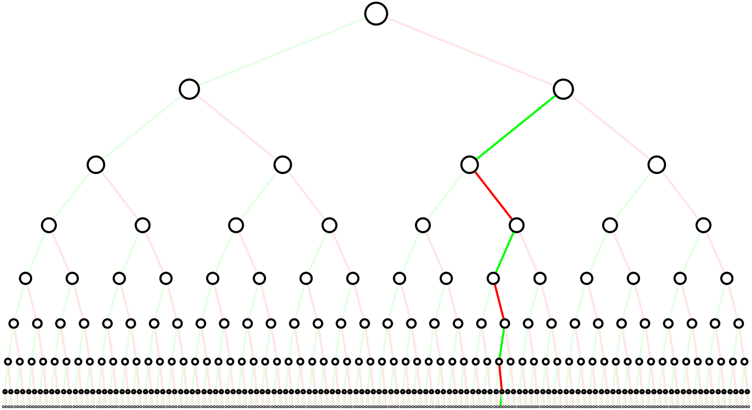
Concepts

pCRA

Bisimulation

Relation between pCRA and DA

Embedding of sequence: — — — — — — — — — — . . .



Puzzle: colouring the infinite binary tree

Problem:
Find a colouring of the tree such that:
Every infinite coloured path is embedded at most once.

Puzzle: colouring the infinite binary tree

Corecursion without Bisimulation

Venanzio Capretta

Colouring Infinite Trees

Concepts

pCRA

Bisimulation

Relation between pCRA and DA

Problem:
Find a colouring of the tree such that:
Every infinite coloured path is embedded at most once.

- Periodic and rational paths cannot be embedded.

Puzzle: colouring the infinite binary tree

Problem:
Find a colouring of the tree such that:
Every infinite coloured path is embedded at most once.

- Periodic and rational paths cannot be embedded.
- Finite number of colours: Same complexity?

Puzzle: colouring the infinite binary tree

Problem:
Find a colouring of the tree such that:
Every infinite coloured path is embedded at most once.

- Periodic and rational paths cannot be embedded.
- Finite number of colours: Same complexity?
- Infinite number of colours: Trivial.

Corecursion without Bisimulation

Venanzio Capretta

Nijmegen, 18 March 2008

Work with

- Tarmo Uustalu, IoC Tallinn and
- Varmo Vene, University of Tartu

The main concepts.

In developing a theory of computation over infinite structures we need the following concepts:

The main concepts.

Corecursion without Bisimulation

Venanzio Capretta

Colouring Infinite Trees

Concepts

pCRA

Bisimulation

Relation between pCRA and DA

In developing a theory of computation over infinite structures we need the following concepts:

- Infinite structures are represented by algebras.

The main concepts.

In developing a theory of computation over infinite structures we need the following concepts:

- Infinite structures are represented by algebras.
- Corecursive algebra: structure to which we can define functions by corecursion;

The main concepts.

In developing a theory of computation over infinite structures we need the following concepts:

- Infinite structures are represented by algebras.
- Corecursive algebra: structure to which we can define functions by corecursion;
- Discriminating algebra: structure that satisfies the principle of bisimulation;

The main concepts.

In developing a theory of computation over infinite structures we need the following concepts:

- Infinite structures are represented by algebras.
- Corecursive algebra: structure to which we can define functions by corecursion;
- Discriminating algebra: structure that satisfies the principle of bisimulation;
- Not equivalent: corecursion doesn't imply bisimulation;

The main concepts.

In developing a theory of computation over infinite structures we need the following concepts:

- Infinite structures are represented by algebras.
- Corecursive algebra: structure to which we can define functions by corecursion;
- Discriminating algebra: structure that satisfies the principle of bisimulation;
- Not equivalent: corecursion doesn't imply bisimulation;
- Counterexample using infinite trees.

Infinite structures as algebras

An algebra for streams: (for the functor $FX = \mathbb{N} \times X^2$)

$$\alpha : \mathbb{N} \times \mathbb{S}^2 \to \mathbb{S}$$
$$\alpha\ (x, \sigma_1, \sigma_2) = x :: (\sigma_1 \oplus \sigma_2)$$

where $\oplus$ is component-wise addition.

Infinite structures as algebras

An algebra for streams: (for the functor $FX = \mathbb{N} \times X^2$)

$$\alpha : \mathbb{N} \times \mathbb{S}^2 \to \mathbb{S}$$
$$\alpha\ (x, \sigma_1, \sigma_2) = x :: (\sigma_1 \oplus \sigma_2)$$

where $\oplus$ is component-wise addition.

Streams are *viewed* as an infinite objects with a (non-unique) tree-like structure.

Infinite structures as algebras

An algebra for streams: (for the functor $FX = \mathbb{N} \times X^2$)

$$\alpha : \mathbb{N} \times \mathbb{S}^2 \to \mathbb{S}$$
$$\alpha\ (x, \sigma_1, \sigma_2) = x :: (\sigma_1 \oplus \sigma_2)$$

where $\oplus$ is component-wise addition.

Streams are *viewed* as an infinite objects with a (non-unique) tree-like structure.

The stream $1 :: 2 :: 3 :: 4 :: \cdots$ is represented by the trees:

Infinite structures as algebras

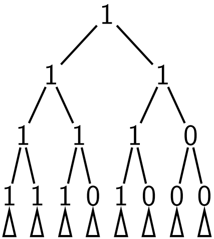
An algebra for streams: (for the functor $FX = \mathbb{N} \times X^2$)

$$\alpha : \mathbb{N} \times \mathbb{S}^2 \to \mathbb{S}$$
$$\alpha\ (x, \sigma_1, \sigma_2) = x :: (\sigma_1 \oplus \sigma_2)$$

where $\oplus$ is component-wise addition.

Streams are *viewed* as an infinite objects with a (non-unique) tree-like structure.

The stream $1 :: 2 :: 3 :: 4 :: \cdots$ is represented by the trees:



Infinite structures as algebras

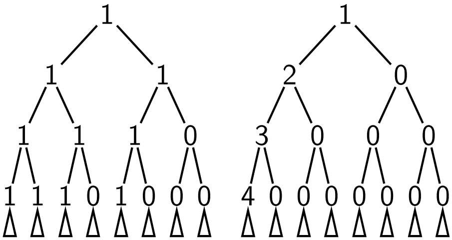
An algebra for streams: (for the functor $FX = \mathbb{N} \times X^2$)

$$\alpha : \mathbb{N} \times \mathbb{S}^2 \rightarrow \mathbb{S}$$
$$\alpha\,(x, \sigma_1, \sigma_2) = x :: (\sigma_1 \oplus \sigma_2)$$

where $\oplus$ is component-wise addition.

Streams are *viewed* as an infinite objects with a (non-unique) tree-like structure.

The stream $1 :: 2 :: 3 :: 4 :: \cdots$ is represented by the trees:



Infinite structures as algebras

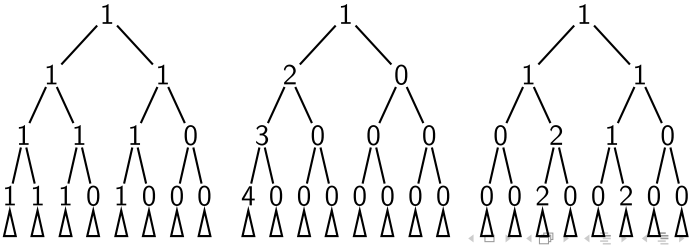
An algebra for streams: (for the functor $FX = \mathbb{N} \times X^2$)

$$\alpha : \mathbb{N} \times \mathbb{S}^2 \to \mathbb{S}$$
$$\alpha\ (x, \sigma_1, \sigma_2) = x :: (\sigma_1 \oplus \sigma_2)$$

where $\oplus$ is component-wise addition.

Streams are *viewed* as an infinite objects with a (non-unique) tree-like structure.

The stream $1 :: 2 :: 3 :: 4 :: \cdots$ is represented by the trees:



Function to the infinite structure

A function exploiting this algebra structure:

$$f : \mathbb{N} \to \mathbb{S}$$
$$f\ x = x :: (f\ (2x) \oplus f\ (2x+1))$$

Function to the infinite structure

A function exploiting this algebra structure:

$$
\begin{aligned}
& f : \mathbb{N} \to \mathbb{S} \\
& f\, x = x :: (f\,(2x) \oplus f\,(2x+1))
\end{aligned}
$$

For example: $f\,1 = 1 :: 5 :: 22 :: 92 :: 376 :: 1520 :: 6112 :: 24512 :: \cdots$.

Function to the infinite structure

A function exploiting this algebra structure:

$$f : \mathbb{N} \to \mathbb{S}$$
$$f\ x = x :: (f\ (2x) \oplus f\ (2x+1))$$

For example: $f\ 1 = 1 :: 5 :: 22 :: 92 :: 376 :: 1520 :: 6112 :: 24512 :: \cdots$.

Decomposition of f: $f = \alpha \circ Ff \circ \beta$

with:

$$\beta : \mathbb{N} \to \mathbb{N} \times \mathbb{N}^2$$
$$\beta\ x = \langle x, 2x, 2x+1 \rangle$$

Corecursion without Bisimulation

Venanzio Capretta

Colouring Infinite Trees

Concepts

pCRA

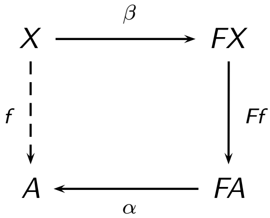
Bisimulation

Relation between pCRA and DA

CoRecursive Algebra

Definition

An algebra $\alpha : A \to FA$ (for a functor F) is called *corecursive (CRA)* if every recursive diagram:

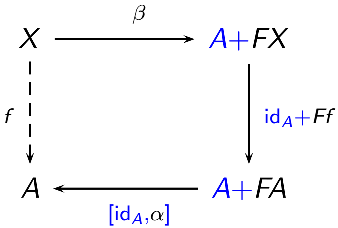


has a unique solution f.

CoRecursive Algebra

Definition

An algebra $\alpha : A \to FA$ (for a functor F)
is called *parametrically corecursive (pCRA)* if
every recursive diagram:



has a unique solution f.

Example of pCRA

Function obtained from a pCRA:

$$
\begin{aligned}
&g : \mathbb{B} \to \mathbb{S} \\
&g\,\mathsf{true} = 1 :: 2 :: 3 :: 4 :: \cdots \\
&g\,\mathsf{false} = 0 :: (g\,\mathsf{true}) \oplus (g\,\mathsf{false})
\end{aligned}
$$

Example of pCRA

Function obtained from a pCRA:

$$
\begin{aligned}
& g : \mathbb{B} \to \mathbb{S} \\
& g\,\mathsf{true} = 1 :: 2 :: 3 :: 4 :: \cdots \\
& g\,\mathsf{false} = 0 :: (g\,\mathsf{true}) \oplus (g\,\mathsf{false})
\end{aligned}
$$

So $g\,\mathsf{false} = 0 :: 1 :: 3 :: 6 :: 10 :: 15 :: 21 :: 28 :: 36 :: 45 :: 55 :: \cdots$.

Example of pCRA

Function obtained from a pCRA:

$$
\begin{array}{l}
g : \mathbb{B} \to \mathbb{S} \\
g\,\mathsf{true} = 1 :: 2 :: 3 :: 4 :: \cdots \\
g\,\mathsf{false} = 0 :: (g\,\mathsf{true}) \oplus (g\,\mathsf{false})
\end{array}
$$

So $g\,\mathsf{false} = 0 :: 1 :: 3 :: 6 :: 10 :: 15 :: 21 :: 28 :: 36 :: 45 :: 55 :: \cdots$.

$$
\begin{array}{l}
\beta : \mathbb{B} \to \mathbb{S} + \mathbb{N} \times \mathbb{B}^2 \\
\beta\,\mathsf{true} = \mathsf{inl}\,(1 :: 2 :: 3 :: 4 :: \cdots) \\
\beta\,\mathsf{false} = \mathsf{inr}\,\langle 0, \mathsf{true}, \mathsf{false} \rangle
\end{array}
$$

pCRAs on the reals

Two examples of pCRAs on the real numbers

pCRAs on the reals

Two examples of pCRAs on the real numbers
First example (Functor $FX = X + X$):

$$\alpha_1 : \mathbb{R} + \mathbb{R} \rightarrow \mathbb{R}$$
$$\alpha_1\,(\mathsf{inl}\,x) = x/2 + 1$$
$$\alpha_1\,(\mathsf{inr}\,x) = x/2 - 1$$

pCRAs on the reals

Two examples of pCRAs on the real numbers
First example (Functor $FX = X + X$):

$$\alpha_1 : \mathbb{R} + \mathbb{R} \to \mathbb{R}$$
$$\alpha_1\,(\mathsf{inl}\,x) = x/2 + 1$$
$$\alpha_1\,(\mathsf{inr}\,x) = x/2 - 1$$

We can see the algebra as a pair of operations:

$$\mathsf{green}, \mathsf{red} : \mathbb{R} \to \mathbb{R}$$
$$\mathsf{green}\,x = x/2 + 1$$
$$\mathsf{red}\,x = x/2 - 1$$

pCRAs on the real

Two examples of pCRAs on the real numbers
First example (Functor $FX = X + X$):

$$\alpha_1 : \mathbb{R} + \mathbb{R} \to \mathbb{R}$$
$$\alpha_1\,(\mathsf{inl}\,x) = x/2 + 1$$
$$\alpha_1\,(\mathsf{inr}\,x) = x/2 - 1$$

We can see the algebra as a pair of operations:

$$\mathsf{green}, \mathsf{red} : \mathbb{R} \to \mathbb{R}$$
$$\mathsf{green}\,x = x/2 + 1$$
$$\mathsf{red}\,x = x/2 - 1$$

Second example (Functor $FX = \mathbb{N} \times X$):

$$\alpha_2 : \mathbb{N} \times \mathbb{R} \to \mathbb{R}$$
$$\alpha_2\,\langle n, x\rangle = n + \arctan x$$

green/red algebras

An algebra for the functor $FX = X + X$ can be seen as a graph with green and red edges

green/red algebras

An algebra for the functor $FX = X + X$ can be seen as a graph with green and red edges

$$(A, \mathsf{green} : A \to A, \mathsf{red} : A \to A)$$

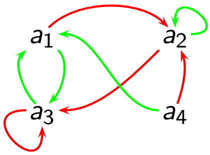
$$\mathsf{green}\, a_1 = a_2, \quad \mathsf{red}\, a_1 = a_3$$
$$\mathsf{green}\, a_2 = a_2, \quad \mathsf{red}\, a_2 = a_3$$
$$\mathsf{green}\, a_3 = a_1, \quad \mathsf{red}\, a_3 = a_3$$
$$\mathsf{green}\, a_4 = a_1, \quad \mathsf{red}\, a_4 = a_2$$

green/red algebras

An algebra for the functor $FX = X + X$ can be seen as a graph with green and red edges

$$(A, \text{green} : A \to A, \text{red} : A \to A)$$

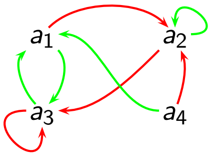
$$\begin{array}{ll} \text{green}\ a_1 = a_2, & \text{red}\ a_1 = a_3 \\ \text{green}\ a_2 = a_2, & \text{red}\ a_2 = a_3 \\ \text{green}\ a_3 = a_1, & \text{red}\ a_3 = a_3 \\ \text{green}\ a_4 = a_1, & \text{red}\ a_4 = a_2 \end{array}$$



green/red algebras

An algebra for the functor $FX = X + X$ can be seen as a graph with green and red edges

$$
\begin{array}{ll}
(A, \text{green} : A \to A, \text{red} : A \to A) & \\
\text{green}\ a_1 = a_2, & \text{red}\ a_1 = a_3 \\
\text{green}\ a_2 = a_2, & \text{red}\ a_2 = a_3 \\
\text{green}\ a_3 = a_1, & \text{red}\ a_3 = a_3 \\
\text{green}\ a_4 = a_1, & \text{red}\ a_4 = a_2
\end{array}
$$

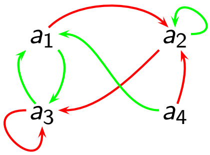


When is an algebra (parametrically) corecursive?

green/red algebras

An algebra for the functor $FX = X + X$ can be seen as a graph with green and red edges

$$(A, \text{green} : A \to A, \text{red} : A \to A)$$
$$\begin{aligned} \text{green } a_1 &= a_2, & \text{red } a_1 &= a_3 \\ \text{green } a_2 &= a_2, & \text{red } a_2 &= a_3 \\ \text{green } a_3 &= a_1, & \text{red } a_3 &= a_3 \\ \text{green } a_4 &= a_1, & \text{red } a_4 &= a_2 \end{aligned}$$



When is an algebra (parametrically) corecursive?

When Every infinite backward red/green sequence of arrows (example: $\leftarrow\leftarrow\leftarrow\leftarrow\leftarrow\leftarrow\cdots$) can be embedded exactly once.

Bisimulation

Informal notion of bisimulation:
Two objects are bisimilar if

- all their observable properties are the same;
- same structure at every level of unfolding.

Bisimulation

Informal notion of bisimulation:
Two objects are bisimilar if

- all their observable properties are the same;
- same structure at every level of unfolding.

For an algebra $(A, \alpha : FA \to A)$: $a_1 \cong a_2$ iff

Bisimulation

Informal notion of bisimulation:
Two objects are bisimilar if

- all their observable properties are the same;
- same structure at every level of unfolding.

For an algebra $(A, \alpha : FA \to A)$: $a_1 \cong a_2$ iff

- they have the same structure and bisimilar components;

Bisimulation

Informal notion of bisimulation:
Two objects are bisimilar if

- all their observable properties are the same;
- same structure at every level of unfolding.

For an algebra $(A, \alpha : FA \rightarrow A)$: $a_1 \cong a_2$ iff

- they have the same structure and bisimilar components;
- if $a_1 = \alpha\, y_1$ and $a_2 = \alpha\, y_2$, then $y_1 \overset{F}{\cong} y_2$.

Bisimulation

Informal notion of bisimulation:
Two objects are bisimilar if

- all their observable properties are the same;
- same structure at every level of unfolding.

For an algebra $(A, \alpha : FA \to A)$: $a_1 \cong a_2$ iff

- they have the same structure and bisimilar components;
- if $a_1 = \alpha\, y_1$ and $a_2 = \alpha\, y_2$, then $y_1 \overset{F}{\cong} y_2$.

$\cong$ is the coarsest relation satisfying these properties.

The next-time operator

Definition (Next-time operator)

$(A, \alpha : FA \to A)$ algebra. Let $\equiv$ be an equivalence on A.

The next time relation $\stackrel{\mathsf{nt}}{\equiv}$ is the reflexive/transitive closure of:

$$\forall y_1, y_2 : FA. \quad y_1 \stackrel{F}{\equiv} y_2 \Rightarrow \alpha\, y_1 \stackrel{\mathsf{nt}}{\equiv} \alpha\, y_2$$

The next-time operator

Definition (Next-time operator)

$(A, \alpha : FA \to A)$ algebra. Let $\equiv$ be an equivalence on A.

The next time relation $\stackrel{\mathsf{nt}}{\equiv}$ is the reflexive/transitive closure of:

$$\forall y_1, y_2 : FA. \quad y_1 \stackrel{F}{\equiv} y_2 \Rightarrow \alpha\, y_1 \stackrel{\mathsf{nt}}{\equiv} \alpha\, y_2$$

In categorical terms:

Given an epimorphism $q : A \twoheadrightarrow Q$ $(Q = A/\equiv)$;

$\mathsf{nt}(Q)$ is defined by a pushout:

$$\begin{array}{ccc} FA & \xrightarrow{\alpha} & A \\ {\scriptstyle Fq}\downarrow & & \downarrow{\scriptstyle \mathsf{nt}(q)} \\ FQ & \xrightarrow[\alpha[q]]{} & \mathsf{nt}(Q) \end{array}$$

green/red bisimulation

Example:

$(A, \mathsf{green} : A \to A, \mathsf{red} : A \to A)$ $(\alpha : A + A \to A)$

$\equiv$ equivalence relation on A

$\overset{\mathsf{nt}}{\equiv}$ is the reflexive/transitive closure of:

$$\forall a_1, a_2 : A. \quad a_1 \equiv a_2 \Rightarrow$$
$$\mathsf{green}\ a_1 \overset{\mathsf{nt}}{\equiv} \mathsf{green}\ a_2 \quad \wedge \quad \mathsf{red}\ a_1 \overset{\mathsf{nt}}{\equiv} \mathsf{red}\ a_2.$$

Discriminating Algebra

Discriminating Algebra (DA):
if two objects are bisimilar, then they are equal.

Discriminating Algebra

Discriminating Algebra (DA):
if two objects are bisimilar, then they are equal.

Definition
(A, α) is a discriminating algebra if
for every equivalence relation $\equiv$ satisfying

$$\forall a_1, a_2. \quad a_1 \equiv a_2 \Rightarrow a_1 \stackrel{\text{nt}}{\equiv} a_2;$$

it must be that $\equiv$ is equality.

Discriminating Algebra

Discriminating Algebra (DA):
if two objects are bisimilar, then they are equal.

Definition
(A, α) is a discriminating algebra if
for every equivalence relation $\equiv$ satisfying

$$\forall a_1, a_2. \quad a_1 \equiv a_2 \Rightarrow a_1 \overset{\mathrm{nt}}{\equiv} a_2;$$

it must be that $\equiv$ is equality.

Categorically: if $\mathrm{nt}(q)$ factorizes through q, then q is iso.

Relation between pCRA and DA

pCRA: every recursive diagram defines a unique object.
DA: structurally similar objects are equal.

Relation between pCRA and DA

pCRA: every recursive diagram defines a unique object.
DA: structurally similar objects are equal.
What is the relation between the two notions?

Relation between pCRA and DA

pCRA: every recursive diagram defines a unique object.
DA: structurally similar objects are equal.
What is the relation between the two notions?

- DA implies pCRA-unicity;

Relation between pCRA and DA

pCRA: every recursive diagram defines a unique object.
DA: structurally similar objects are equal.
What is the relation between the two notions?

- DA implies pCRA-unicity;
- DA does not imply pCRA-existence;

Relation between pCRA and DA

pCRA: every recursive diagram defines a unique object.
DA: structurally similar objects are equal.
What is the relation between the two notions?

- DA implies pCRA-unicity;
- DA does not imply pCRA-existence;
- pCRA does not imply DA.

Relation between pCRA and DA

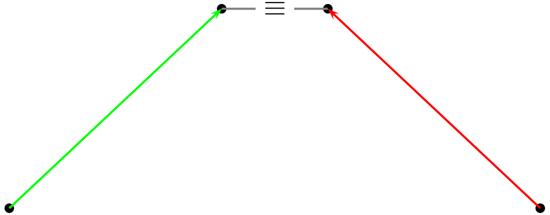
pCRA: every recursive diagram defines a unique object.
DA: structurally similar objects are equal.
What is the relation between the two notions?

- DA implies pCRA-unicity;
- DA does not imply pCRA-existence;
- pCRA does not imply DA.

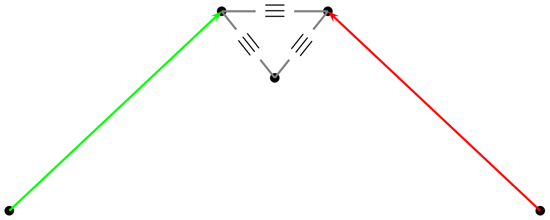
Counterexample exploiting transitive closure.

pCRA but not DA

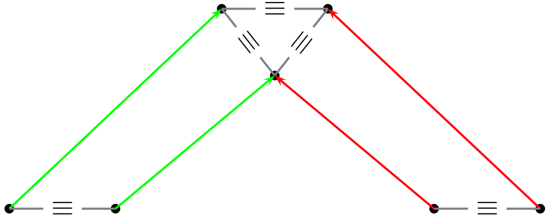
pCRA but not DA



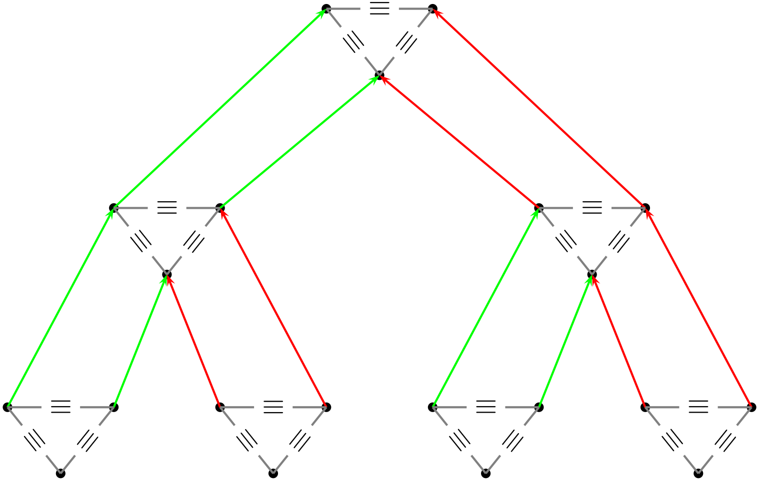
pCRA but not DA



pCRA but not DA



pCRA but not DA



The Counterexample

Actual counterexample:

- Algebra for the functor $FX = \mathbb{N} \times X$;
- Satisfies pCRA but not DA;
- Contains an infinite tree:
  - Each node contains three distinct elements;
  - The three elements are bisimilar;
  - Inherent use of transitive closure.

Formal details:
Venanzio Capretta, Tarmo Uustalu, and Varmo Vene,
Corecursive Algebras: A Study of General Structured Corecursion