Syntactic Theory of Finitary Sets

Non well-founded sets

Denis Müller

Saarland University

10 April 2015

Table of Contents

Introduction

Non-well-founded sets

- may contain cycles, e.g. $M \in M$
- can be represented by rooted graphs up to bisimulation
- Our Model : ZF - Regularity - Infinity + AFA

Representation

Consider graphs over a type X:
Inductive Graph X := $G(xs : \mathit{list}\, X)(f : X \to X \to \mathbb{B})(r : X)$

Reachability in a graph :

$$\frac{}{x \underset{\mathsf{f}}{\to}^{*} x}$$

$$\frac{x \underset{\mathsf{f}}{\to}^{*} y \qquad f\ y\ z = \mathit{true}}{x \underset{\mathsf{f}}{\to}^{*} z}$$

Representation

Non well-founded set $\approx$ rooted graph over X.

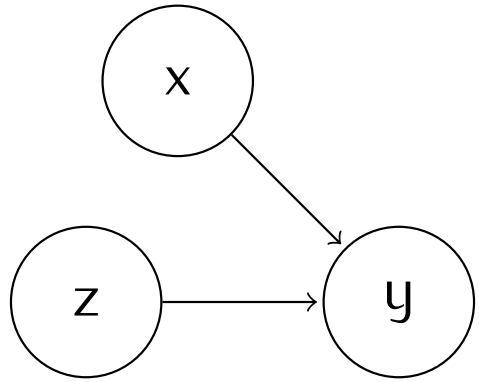
valid (G xs f r) :=

$$(\forall x\, y, f\, x\, y = \mathit{true} \implies x \in xs \wedge y \in xs) \wedge$$
$$r \in xs \wedge \forall x \in xs. r \xrightarrow[\mathrm{f}]{}{}^{*} x)$$

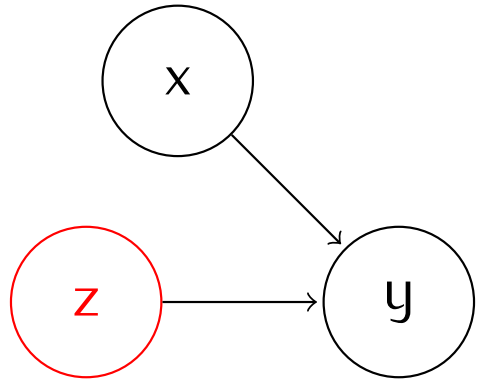
NSet := sig valid.

Non well-founded sets := $\mathit{NSet}\,/_{\approx}$

Examples

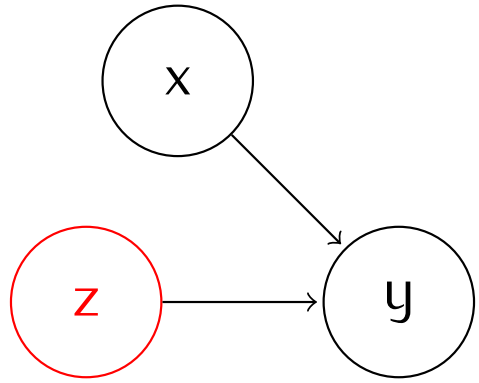


Examples

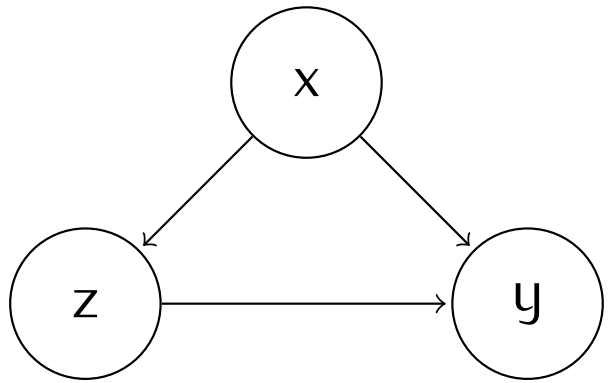


not valid.

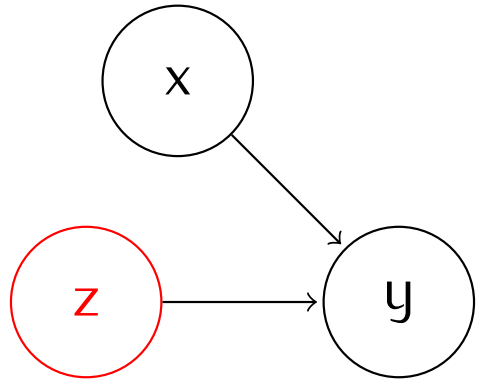
Examples



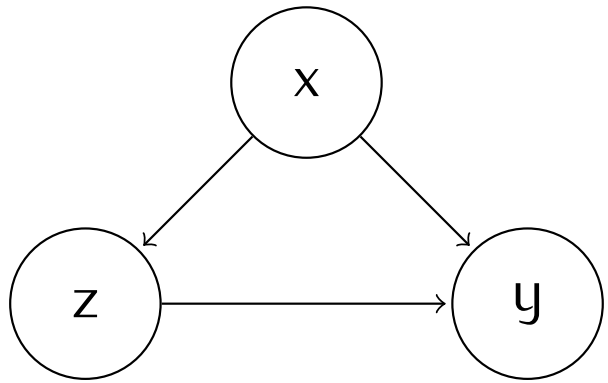
not valid.



Examples



not valid.



valid.

Table of Contents

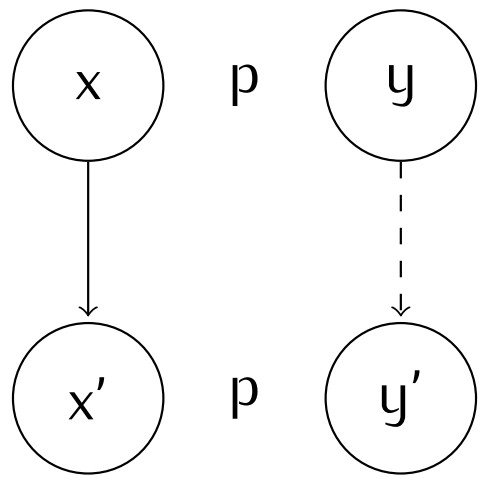
Table of Contents

Bisimulation

Natural equivalence relation on non well-founded sets: bisimulation.
p : $X \to Y \to \mathbb{B}$ is a bisimulation for
((G xs f r) : Graph X) ((G ys g q) : Graph Y) :=

Bisimulation

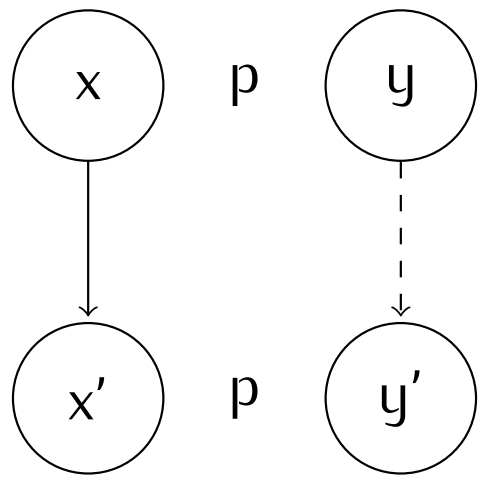
Natural equivalence relation on non well-founded sets: bisimulation.
p : $X \to Y \to \mathbb{B}$ is a bisimulation for
((G xs f r) : Graph X) ((G ys g q) : Graph Y) :=



(and vice-versa)

Bisimulation

Natural equivalence relation on non well-founded sets: bisimulation.
p : $X \to Y \to \mathbb{B}$ is a bisimulation for
((G xs f r) : Graph X) ((G ys g q) : Graph Y) :=



(and vice-versa)

(G xs f r) $\approx$ (G ys g q) := exists p,
p is a bisimulation for (G xs f r) , (G ys g q) $\wedge$
p r q = true.

Table of Contents

Decide if p is a bisimulation

- transition functions limited to node list (assuming validity)
- test all - finitely many - nodes
- use list exists and forall decidability

Decide if p is a bisimulation

- transition functions limited to node list (assuming validity)
- test all - finitely many - nodes
- use list exists and forall decidability

$$
\begin{aligned}
dec((\forall x, x \in xs \implies \forall y, y \in ys \implies p\, x\, y = true \implies \\
(\forall x', x' \in xs \implies f\, x\, x' = true \implies \\
\exists y', y' \in ys \wedge g\, y\, y' = true \wedge P\, x'\, y' = true) \wedge \\
(\forall y', y' \in ys \implies g\, y\, y' = true \implies \\
\exists x', x' \in xs \wedge f\, x\, x' = true \wedge P\, x'\, y' = true)))
\end{aligned}
$$

Decide if p is a bisimulation

- transition functions limited to node list (assuming validity)
- test all - finitely many - nodes
- use list exists and forall decidability

$$dec((\forall x, x \in xs \implies \forall y, y \in ys \implies p\,x\,y = true \implies (\forall x', x' \in xs \implies f\,x\,x' = true \implies \exists y', y' \in ys \wedge g\,y\,y' = true \wedge P\,x'\,y' = true) \wedge (\forall y', y' \in ys \implies g\,y\,y' = true \implies \exists x', x' \in xs \wedge f\,x\,x' = true \wedge P\,x'\,y' = true)))$$

Proof. auto. Qed.

Decide if $s \approx t$

How to decide if (G xs f r) $\approx$ (G ys g q)

Decide if $s \approx t$

How to decide if (G xs f r) $\approx$ (G ys g q)

- only finitely many relations on xs $\times$ ys

Decide if $s \approx t$

How to decide if (G xs f r) $\approx$ (G ys g q)

- only finitely many relations on xs $\times$ ys
- $map(\lambda A.\lambda x\, y.(x, y) \in A)(\mathcal{P}(xs \times ys)\ )$

Decide if $s \approx t$

How to decide if (G xs f r) $\approx$ (G ys g q)

- only finitely many relations on xs $\times$ ys
- $map(\lambda A.\lambda x\, y.(x, y) \in A)(\mathcal{P}(xs \times ys)\ )$
- Check for a bisimulation among these relations (1 slide ago)

Table of Contents

Table of Contents

Reachability

Goal : decide if a node x reaches a node y in a Graph (G xs f r).
Definition of $x \xrightarrow[f]{}{}^{*} y$ rather unsuitable for decidability.
Use different characterizations :

- reachability in n steps ($x \xrightarrow[f]{}{}^{n} y$) obviously decidable
- explicit path as list

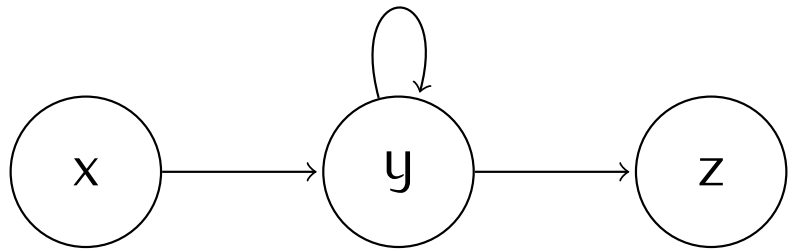
Explicit Path

Represent path as list of nodes

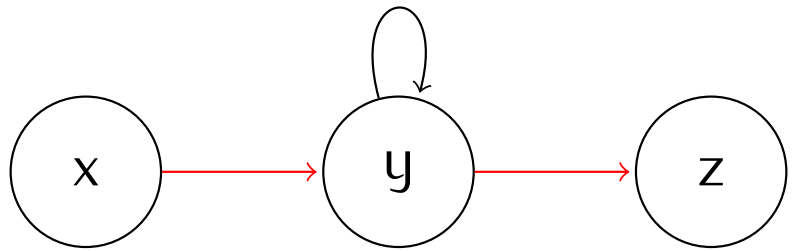
$$\frac{}{[x] : x \underset{f}{\rightarrow}^{*} x}$$

$$\frac{x \underset{f}{\rightarrow} y \qquad xs : y \underset{f}{\rightarrow}^{*} z}{(x :: xs) : x \underset{f}{\rightarrow}^{*} z}$$

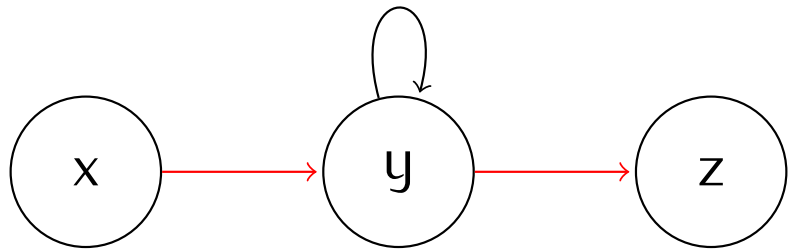
Paths without loops



Paths without loops

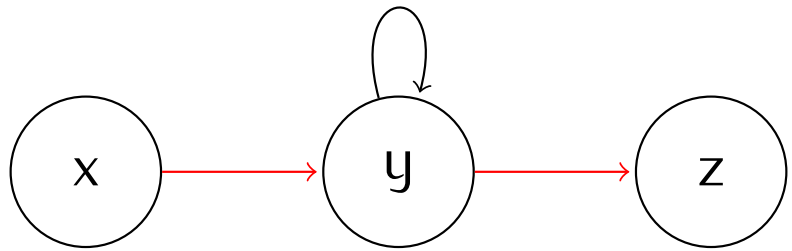


Paths without loops



ρ [] := []
ρ (x :: xs) :=
    let ys := ρ xs in
        if (x $\in$ ys) then remove_until x ys else x :: ys

Paths without loops



ρ [] := []
ρ (x :: xs) :=
    let ys := ρ xs in
        if (x $\in$ ys) then remove_until x ys else x :: ys

remove_until x [] := []
remove_until x (y :: ys) :=
    if (x = y) then y :: ys else remove_until x ys

Properties of ρ

- $\mathsf{xs} : \mathsf{x} \underset{\mathsf{f}}{\rightarrow}^{*} \mathsf{y} \implies \rho\ \mathsf{xs} : \mathsf{x} \underset{\mathsf{f}}{\rightarrow}^{*} \mathsf{y}$
- ρ xs does not contain duplicates
- $\mathsf{xs} \subseteq \mathsf{ys} \implies |\rho\ \mathsf{xs}| \leq |\mathsf{ys}|$

Path decidability

Decide if there is a path from x to y in (G xs f r) , i.e. $x \underset{f}{\rightarrow}^{*} y$.

- Check if ($x \underset{f}{\rightarrow}^{|xs|} y$)

Path decidability

Decide if there is a path from x to y in (G xs f r) , i.e. $x \underset{f}{\rightarrow}^{*} y$.

- Check if $(x \underset{f}{\rightarrow}^{|xs|} y)$
- If true, we have a path

Path decidability

Decide if there is a path from x to y in (G xs f r) , i.e. $x \underset{f}{\rightarrow}^{*} y$.

- Check if ($x \underset{f}{\rightarrow}^{|xs|} y$)
- If true, we have a path
- If false, there is no path: (assume $x \underset{f}{\rightarrow}^{*} y$)

Path decidability

Decide if there is a path from x to y in (G xs f r) , i.e. $x \underset{f}{\rightarrow}^{*} y$.

- Check if ($x \underset{f}{\rightarrow}^{|xs|} y$)
- If true, we have a path
- If false, there is no path: (assume $x \underset{f}{\rightarrow}^{*} y$)
  - If $x \underset{f}{\rightarrow}^{*} y$, then there is a path ys from x to y

Path decidability

Decide if there is a path from x to y in (G xs f r) , i.e. $x \underset{f}{\rightarrow}^{*} y$.

- Check if ($x \underset{f}{\rightarrow}^{|xs|} y$)
- If true, we have a path
- If false, there is no path: (assume $x \underset{f}{\rightarrow}^{*} y$)
  - If $x \underset{f}{\rightarrow}^{*} y$, then there is a path ys from x to y
  - Hence, ρ ys is also a path from x to y

Path decidability

Decide if there is a path from x to y in (G xs f r) , i.e. $x \underset{f}{\rightarrow}^{*} y$.

- Check if ($x \underset{f}{\rightarrow}^{|xs|} y$)
- If true, we have a path
- If false, there is no path: (assume $x \underset{f}{\rightarrow}^{*} y$)
  - If $x \underset{f}{\rightarrow}^{*} y$, then there is a path ys from x to y
  - Hence, ρ ys is also a path from x to y
  - $|\rho\, ys| \leq |xs|$, since $ys \subseteq xs$

Path decidability

Decide if there is a path from x to y in (G xs f r) , i.e. $x \underset{f}{\rightarrow}^{*} y$.

- Check if ($x \underset{f}{\rightarrow}^{|xs|} y$)
- If true, we have a path
- If false, there is no path: (assume $x \underset{f}{\rightarrow}^{*} y$)
  - If $x \underset{f}{\rightarrow}^{*} y$, then there is a path ys from x to y
  - Hence, ρ ys is also a path from x to y
  - $|\rho\, ys| \leq |xs|$, since $ys \subseteq xs$
  - Hence, $x \underset{f}{\rightarrow}^{|xs|} y = \text{true}$, contradiction

Table of Contents

Subgraphs

Limit transition function on a list of nodes :
trim f xs := $\lambda xy.\mathit{if}\, x \in xs \wedge y \in xs$ then f x y else false

Subgraphs

Limit transition function on a list of nodes :
trim f xs := $\lambda xy.\mathit{if}\, x \in xs \wedge y \in xs$ then f x y else false

Subgraph reachable from a node x :

- ys := all nodes reachable from x
- g := trim f ys
- root of subgraph : x

Subgraphs

Limit transition function on a list of nodes :
trim f xs := $\lambda xy.\mathit{if}\, x \in xs \wedge y \in xs$ then f x y else false

Subgraph reachable from a node x :

- ys := all nodes reachable from x
- g := trim f ys
- root of subgraph : x

Subgraph for x is always valid (provided x ∈ xs).
children : Graph X → list (Graph X)

Element relation

Graph element relation :

$$s \in t := \exists\, x \in \text{children } t.\ s \approx x$$

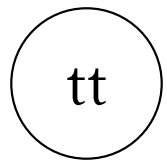
Same for NSet : child-sets, element relation

Future work : $M \approx N \iff (\forall\, x.\ x \in M \iff x \in N)$.

Table of Contents

Empty

$\forall$ u, u $\notin$ empty x.
empty graph : Graph Unit
Inductive Unit := tt.
empty graph := G [tt] ($\lambda xy.false$) tt



From lists to Graphs

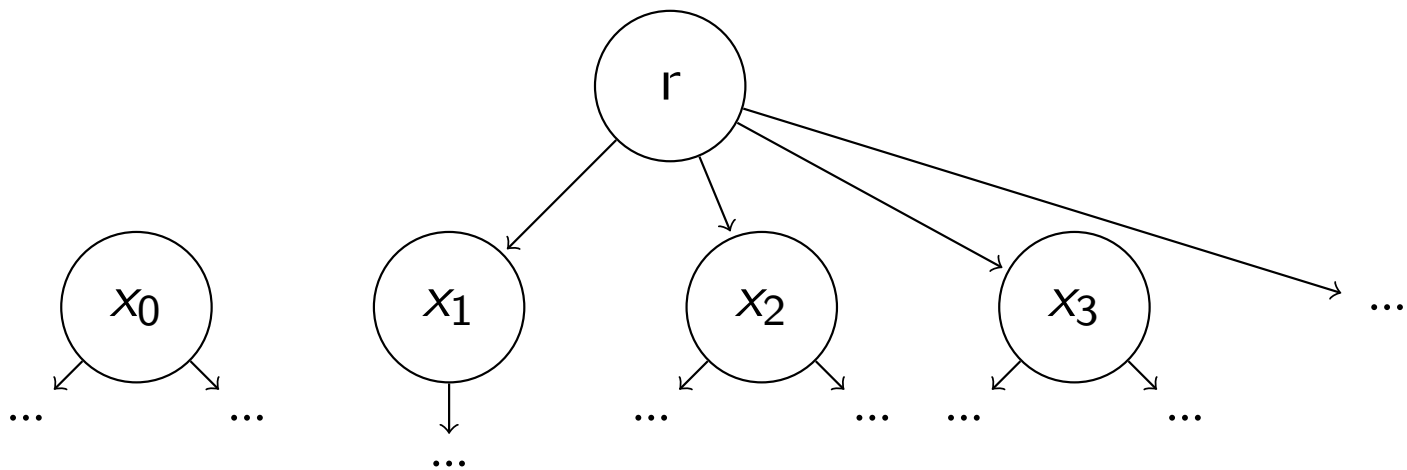
list_to_graph : list Graph → Graph

- [] ⇒ empty graph
- (x :: xs) ⇒ add x as a child to list_to_graph xs

From lists to Graphs

list_to_graph : list Graph → Graph

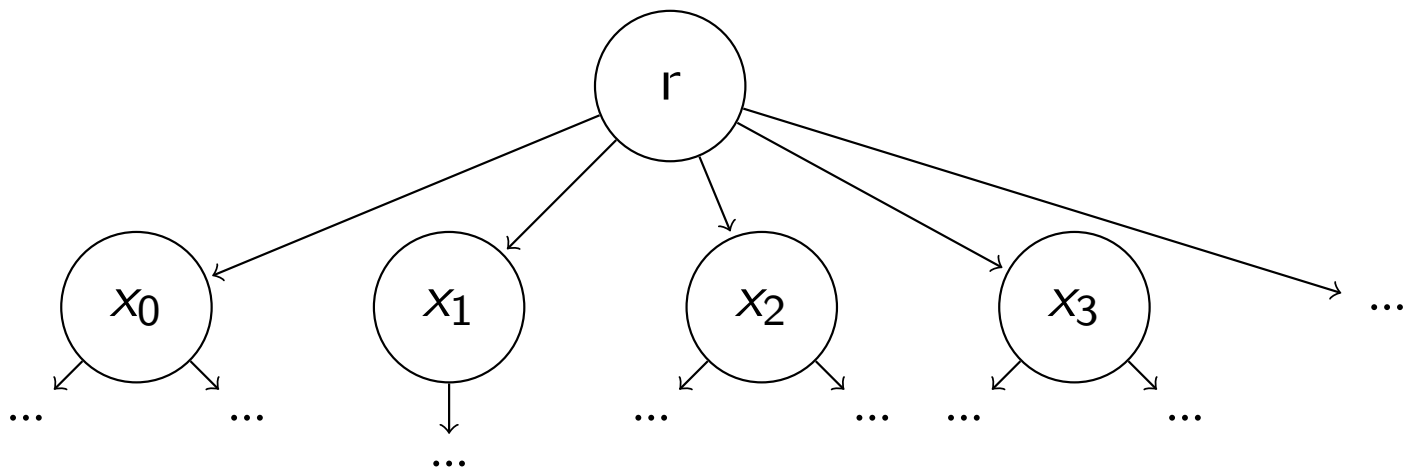
- [] ⇒ empty graph
- (x :: xs) ⇒ add x as a child to list_to_graph xs



From lists to Graphs

list_to_graph : list Graph → Graph

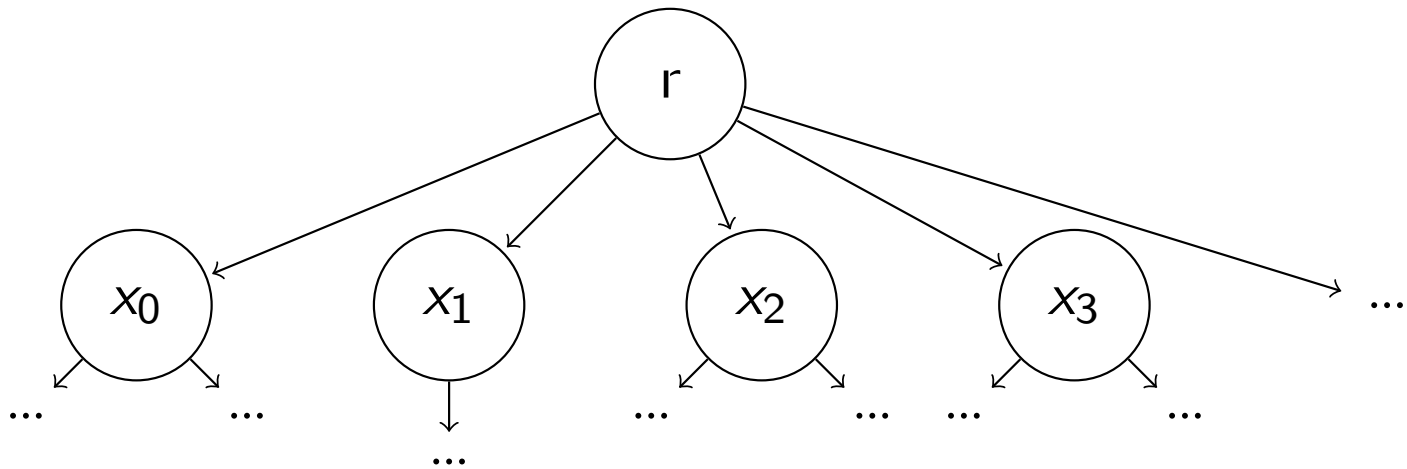
- [] ⇒ empty graph
- (x :: xs) ⇒ add x as a child to list_to_graph xs



From lists to Graphs

list_to_graph : list Graph → Graph

- [] ⇒ empty graph
- (x :: xs) ⇒ add x as a child to list_to_graph xs



Same for list_to_set : list NSet → NSet

NSets and Lists

NSet X → list NSet:

- Take the children of the underlying graph
- Convert children to NSets (all children are valid)

list NSet → NSet

- Transform list of Graphs to Graph
- Validity is preserved

Other Axioms

Known : Empty, Conversions NSet $\iff$ list NSet

Other Axioms

Known : Empty, Conversions NSet $\iff$ list NSet
Missing ZF-Axioms : Upair, Union, Adjunction, Separation, Replacement, Power
(no Regularity or Infinity here)

Other Axioms

Known : Empty, Conversions NSet $\iff$ list NSet
Missing ZF-Axioms : Upair, Union, Adjunction, Separation, Replacement, Power
(no Regularity or Infinity here)
Idea : convert to list, transform list, convert back

Upair, Singleton

$N' \in \{N, M\} \iff N' \approx N \vee N' \approx M$

- Take xs := [N,M] : list NSet
- Transform to NSet

Upair, Singleton

$N' \in \{N, M\} \iff N' \approx N \vee N' \approx M$

- Take xs := [N,M] : list NSet
- Transform to NSet

$N \in \{M\} \iff N \approx M.$
$\{M\} := \{M, M\}$

Union

$$N \in \cup M \iff \exists M' \in M.\ N \in M'.$$

$$\cup M := \text{list\_to\_set (flatten (map child\_sets (child\_sets M)))}$$

Adjunction

$$N' \in N; M \iff N' \in N \vee N' \approx M$$

$$\mathsf{N};\mathsf{M} := \cup\{N, \{M\}\}$$

Separation

$$\mathsf{P} : \mathbb{N}\mathsf{Set} \to \mathsf{Prop}$$

$$N \in \{M' \in M | P\, M'\} \iff \exists M' \in M.P\, M' \wedge N \approx M'$$

$$\{M' \in M | P\, M'\} := \mathsf{list\_to\_set}\ (\ \mathsf{filter}\ \mathsf{P}\ (\mathsf{child\_sets}\ \mathsf{M}))$$

$$\mathsf{P}\ \text{decidable},\ \mathsf{P}\ \text{extensional}$$

Replacement

$$\mathsf{f} : \mathbb{N}\mathsf{Set} \to \mathbb{N}\mathsf{Set}$$
$$N \in \{f\ M' | M' \in M\} \iff \exists M' \in M. N \approx f\ M'$$
$$\{f\ M' | M' \in M\} := \mathsf{list\_to\_set}\ (\mathsf{map}\ \mathsf{f}\ (\mathsf{child\_sets}\ \mathsf{M}))$$
$$\forall\ \mathsf{M}\ \mathsf{N}.\ \mathsf{M} \approx \mathsf{N} \to \mathsf{f}\ \mathsf{N} \approx \mathsf{f}\ \mathsf{M}.$$

Power

$$\mathsf{N} \in \mathcal{P}(M) \iff \mathsf{N} \subseteq \mathsf{M}.$$

$$\mathcal{P}(M) := \mathsf{list\_to\_set}\ (\mathsf{map}\ \mathsf{list\_to\_set}\ (\mathcal{P}\ (\mathsf{child\_sets}\ \mathsf{M})))$$

Table of Contents

Anti-Foundation Axiom

Azcel : Every non well-founded set that has an apg (accessible pointed graph) exists.
Our model for non well-founded sets : apgs
Only restriction : transition function.

Anti-Foundation Axiom

Azcel : Every non well-founded set that has an apg (accessible pointed graph) exists.
Our model for non well-founded sets : apgs
Only restriction : transition function.
AFA : $\forall xs\ f\ x.x \in xs \implies$
$(\exists ys\ g.valid(G\ ys\ g\ x) \wedge$
$(\forall a\ b.g\ a\ b = true \iff f\ a\ b = true \wedge a \xrightarrow[\text{trim f xs}]{}{}^{*}\ b))$

Anti-Foundation Axiom

Azcel : Every non well-founded set that has an apg (accessible pointed graph) exists.
Our model for non well-founded sets : apgs
Only restriction : transition function.
AFA : $\forall xs\ f\ x.x \in xs \implies$
$(\exists ys\ g.valid(G\ ys\ g\ x) \wedge$
$(\forall a\ b.g\ a\ b = true \iff f\ a\ b = true \wedge a \xrightarrow[\text{trim f xs}]{}{}^{*}\ b))$
AFA xs f r := subgraph (G xs f r) r

Table of Contents

References

- Peter Aczel. Non-Well-Founded Sets. CSLI Lecture Notes Vol. 14. Stanford University, 1988.
- S. Abramski. A Cook's Tour of the Finitary Non-Well-Founded Sets, 2011.
- Sangiorgi, Davide. Introduction to bisimulation and coinduction. Cambridge University Press, 2011.
- Kathrin Stark. Quantitative Recursion-Free Process Axiomatization in Coq, 2014.