A Syntactic Theory Of Finitary Sets

Denis Müller

Saarland University

31.7.15

Overview

Table of Contents

Non-well-founded sets

We give a constructive model for non-well-founded sets.

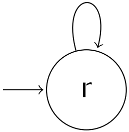
- Our Model : ZFC - Regularity - Infinity + AFA
- Non-well-founded sets can be represented by rooted graphs up to bisimulation

Non-well-founded sets

We give a constructive model for non-well-founded sets.

- Our Model : ZFC - Regularity - Infinity + AFA
- Non-well-founded sets can be represented by rooted graphs up to bisimulation

Simplest example: $\Omega = \{\Omega\}$



Graphs

Definition

A (rooted) graph is a 4-tuple (X, edgeRel, dom, root), where

- *X is a type with decidable equality*
- *edgeRel :* $X \to X \to \mathbb{B}$ *is the transition relation*
- *dom : [X] is the domain of the graph*
- *root : X denotes the root of the graph*

We denote the type associated with a graph g by t g (or simply g) and the type of graphs by $\mathbb{G}$.

Edges

We always only consider the subgraph induced by the domain, hence the following definition:

Definition

$$E : \; g \to g \to \mathbb{B}$$

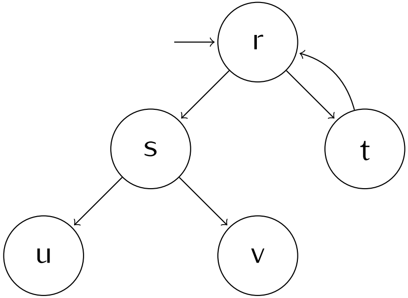
$$E\ x\ y := \begin{cases} edgeRel\ x\ y & x \in dom(g) \land y \in dom(g) \\ false & otherwise \end{cases}$$

Child nodes

Child nodes : reachable from the root in one step.

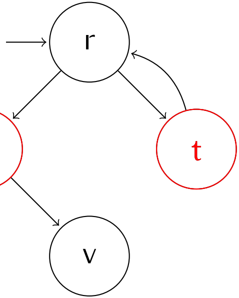
Child nodes

Child nodes : reachable from the root in one step.



Child nodes

Child nodes : reachable from the root in one step.

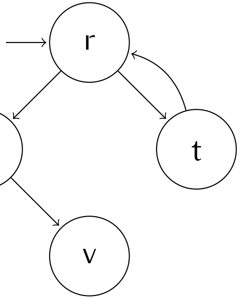


Subgraphs

The subgraph for a vertex x is obtained by setting the root to x:

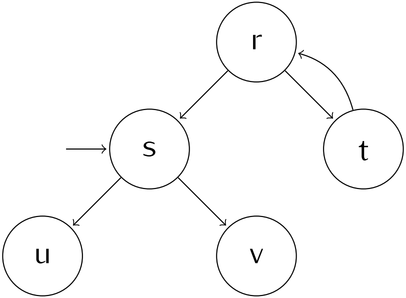
Subgraphs

The subgraph for a vertex x is obtained by setting the root to x:



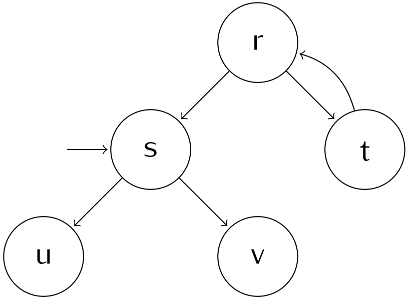
Subgraphs

The subgraph for a vertex x is obtained by setting the root to x:



Subgraphs

The subgraph for a vertex x is obtained by setting the root to x:



The children of a graph denote the subgraphs starting from its child nodes.

Bisimulation

A relation $p : \mathbb{G} \to \mathbb{G} \to \mathbb{B}$ is a bisimulation (**bisim p**) if

$$\begin{tikzcd} x \arrow[r, phantom, "p"] \arrow[d] & y \arrow[d, dashed] \\ x' \arrow[r, phantom, "p"] & y' \end{tikzcd}$$

(and vice-versa)

Bisimulation

A relation p : $\mathbb{G} \to \mathbb{G} \to \mathbb{B}$ is a bisimulation (**bisim p**) if

$$\begin{tikzcd} x \arrow[r, phantom, "p"] \arrow[d] & y \arrow[d, dashed] \\ x' \arrow[r, phantom, "p"] & y' \end{tikzcd}$$

(and vice-versa)

Two graphs g_1, g_2 are bisimilar ($g_1 \approx g_2$) if
$\exists p.\ bisim\ p \wedge p\ (root\ g_1)\ (root\ g_2)$.

Elements and subsets

Based on $\approx$ and the children of a graph, we can define an element relation:

Definition

$g_1 \mathrel{\dot{\in}} g_2 := \exists g \in \mathit{children} g_2 .\ g_1 \approx g.$

Elements and subsets

Based on $\approx$ and the children of a graph, we can define an element relation:

Definition

$g_1 \mathrel{\dot{\in}} g_2 := \exists g \in \mathit{children} g_2 .\, g_1 \approx g.$

Definition

$g_1 \mathrel{\dot{\subseteq}} g_2 := \forall g .\, g \mathrel{\dot{\in}} g_1 \implies g \mathrel{\dot{\in}} g_2.$

Definition

$g_1 \equiv g_2 := g_1 \mathrel{\dot{\subseteq}} g_2 \land g_2 \mathrel{\dot{\subseteq}} g_1.$

Outline

Already known:

- How to decide $g_1 \approx g_2$, $g_1 \dot{\in} g_2$, $g_1 \dot{\subseteq} g_2$, $g_1 \equiv g_2$.
- How to decide reachability in a graph, i.e. $x \rightarrow^* y$.
- Constructions for all ZF axioms except Infinity, Regularity and Extensionality.

Outline

Already known:

- How to decide $g_1 \approx g_2$, $g_1 \mathrel{\dot{\in}} g_2$, $g_1 \mathrel{\dot{\subseteq}} g_2$, $g_1 \equiv g_2$.
- How to decide reachability in a graph, i.e. $x \rightarrow^* y$.
- Constructions for all ZF axioms except Infinity, Regularity and Extensionality.

Today:

- Extensionality
- Transitive closure
- Quotient type
- Choice function

Table of Contents

Extensionality

The only missing ZF axiom that is admissible for our model is extensionality:

Theorem (Extensionality)

$\forall g_1\, g_2.\, g_1 \approx g_2 \iff g_1 \equiv g_2.$

Extensionality "⇒"

"⇒" Let $g_1 \approx g_2$. We show $g_1 \dot{\subseteq} g_2$.

Extensionality "⇒"

"⇒" Let $g_1 \approx g_2$. We show $g_1 \dot{\subseteq} g_2$.

- Let $g \dot{\in} g_1$, i.e. there is a vertex $x \in \mathit{dom}\, g_1$ such that $E(\mathit{root}\, g_1)x = \mathit{true} \wedge g \approx \mathit{subgraph}\, x$.

Extensionality "⇒"

"⇒" Let $g_1 \approx g_2$. We show $g_1 \dot{\subseteq} g_2$.

- Let $g \dot{\in} g_1$, i.e. there is a vertex $x \in dom\, g_1$ such that $E(root\, g_1)x = true \wedge g \approx subgraph\, x$.
- Since $g_1 \approx g_2$, there is some vertex $y \in dom\, g_2$ such that $E\,(root\, g_2)\, y = true \wedge p\, x\, y = true$, where p is the witness of $g_1 \approx g_2$.

Extensionality "⇒"

"⇒" Let $g_1 \approx g_2$. We show $g_1 \dot{\subseteq} g_2$.

- Let $g \dot{\in} g_1$, i.e. there is a vertex $x \in dom\, g_1$ such that $E(root\, g_1)x = true \land g \approx subgraph\, x$.
- Since $g_1 \approx g_2$, there is some vertex $y \in dom\, g_2$ such that $E\,(root\, g_2)\, y = true \land p\, x\, y = true$, where p is the witness of $g_1 \approx g_2$.
- To show $g \approx subgraph\, y$, it suffices to show $subgraph\, x \approx subgraph\, y$. It is easy to see that the relation p is also a bisimulation for subgraph x and subgraph y.

Extensionality "$\Leftarrow$"

"$\Leftarrow$" Let $g_1 \equiv g_2$ and $\mathrm{p} := \lambda x\, y.\, \mathit{subgraph}\, x \equiv \mathit{subgraph}\, y$.

Extensionality "$\Leftarrow$"

"$\Leftarrow$" Let $g_1 \equiv g_2$ and p $:= \lambda x\, y.\, subgraph\, x \equiv subgraph\, y$.

- Obviously, p (root g_1) (root g_2) $=$ true.

Extensionality "$\Leftarrow$"

"$\Leftarrow$" Let $g_1 \equiv g_2$ and $\mathrm{p} := \lambda x\, y.\, subgraph\, x \equiv subgraph\, y$.

- Obviously, p (root g_1) (root g_2) $=$ true.
- Consider $x, x' \in dom\, g_1$ such that $E\, x\, x' = true$, $y \in dom\, g_2$ and $\mathrm{p}\ \mathrm{x}\ \mathrm{y} =$ true.

Extensionality "$\Leftarrow$"

"$\Leftarrow$" Let $g_1 \equiv g_2$ and $\mathrm{p} := \lambda x\, y.\, \mathit{subgraph}\, x \equiv \mathit{subgraph}\, y$.

- Obviously, p (root g_1) (root g_2) $=$ true.
- Consider $x, x' \in \mathit{dom}\, g_1$ such that $E\, x\, x' = \mathit{true}$, $y \in \mathit{dom}\, g_2$ and $\mathrm{p\ x\ y} =$ true.
- Since $\mathit{subgraph}\, x \equiv \mathit{subgraph}\, y$, there is some $y' \in \mathit{dom}\, g_2$ such that $E\, y\, y' = \mathit{true} \wedge \mathit{subgraph}\, x' \approx \mathit{subgraph}\, y'$.

Extensionality "$\Leftarrow$"

"$\Leftarrow$" Let $g_1 \equiv g_2$ and $\mathrm{p} := \lambda x\, y.\, \mathit{subgraph}\, x \equiv \mathit{subgraph}\, y$.

- Obviously, p (root g_1) (root g_2) = true.
- Consider $x, x' \in \mathit{dom}\, g_1$ such that $E\, x\, x' = \mathit{true}$, $y \in \mathit{dom}\, g_2$ and p x y = true.
- Since $\mathit{subgraph}\, x \equiv \mathit{subgraph}\, y$, there is some $y' \in \mathit{dom}\, g_2$ such that $E\, y\, y' = \mathit{true} \wedge \mathit{subgraph}\, x' \approx \mathit{subgraph}\, y'$.
- Due to the direction already proven, we know that $\mathit{subgraph}\, x' \equiv \mathit{subgraph}\, y'$.

Table of Contents

Transitive closure

The transitive closure of a set is basically the set of all its successors w.r.t. the element relation.
Its usual definition relies on the axiom of infinity:
$tc\, M := \bigcup_{n \in \mathbb{N}} (\bigcup^n M)$

Transitive closure

Successors of a graph (w.r.t $\dot{\in}$) correspond to vertices that are reachable from its root.

Transitive closure

Successors of a graph (w.r.t $\dot{\in}$) correspond to vertices that are reachable from its root.

Definition

$x \rightarrow^{+} y := \exists x'.\, E\,x\,x' = \mathit{true} \wedge x' \rightarrow^{*} y.$

Transitive closure

Successors of a graph (w.r.t $\dot{\in}$) correspond to vertices that are reachable from its root.

Definition

$x \rightarrow^{+} y := \exists x'.\, E\, x\, x' = \mathit{true} \wedge x' \rightarrow^{*} y.$

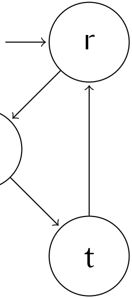
$x \rightarrow^{+} y$ is obviously decidable.

Transitive closure

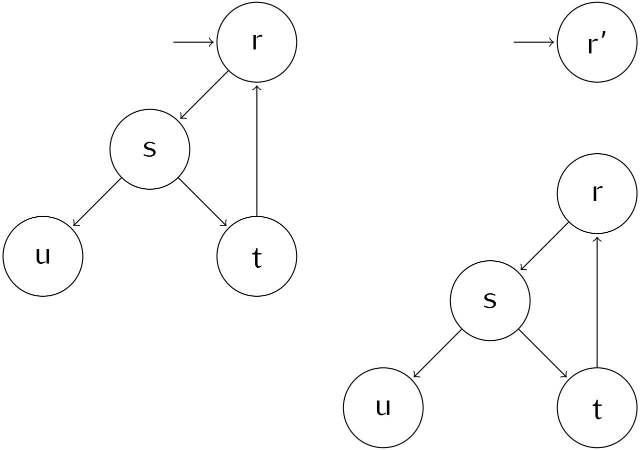
The construction works as follows:

- add a new root
- make the new root adjacent to every vertex v such that root g $\rightarrow^{+}$ v

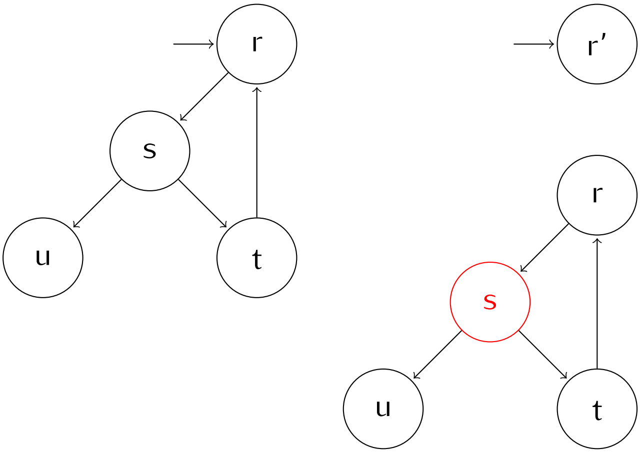
Example



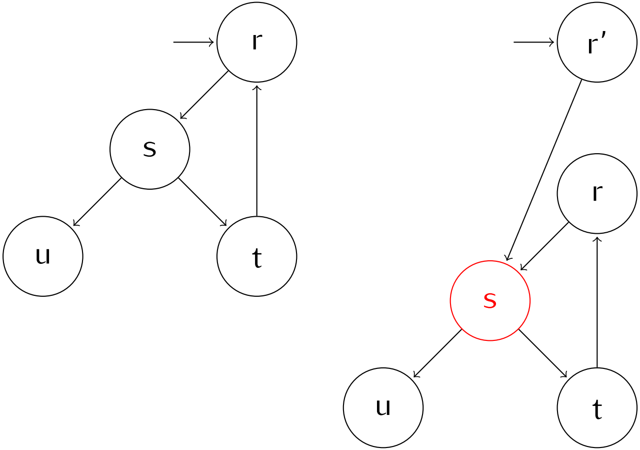
Example



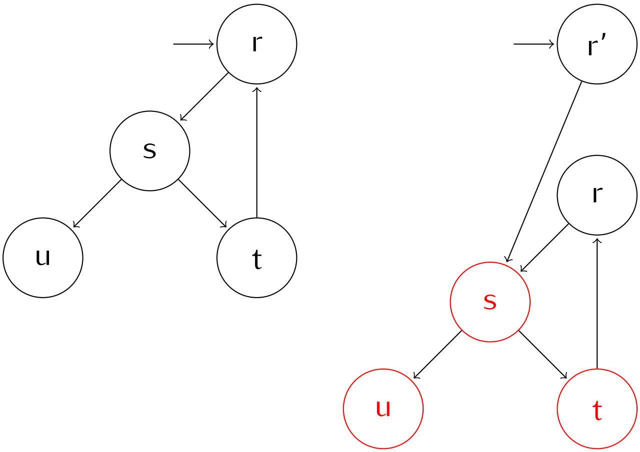
Example



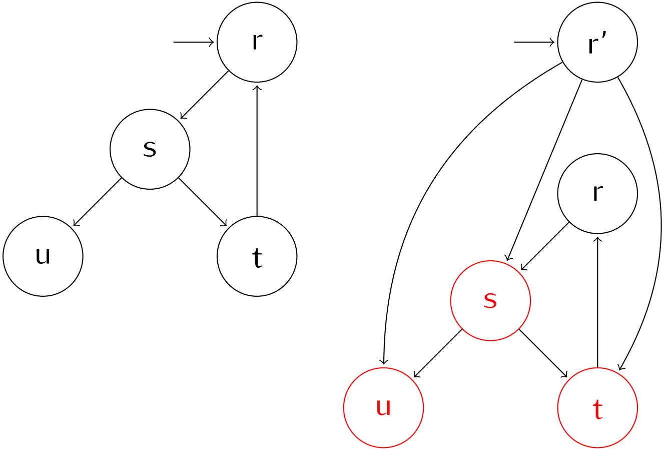
Example



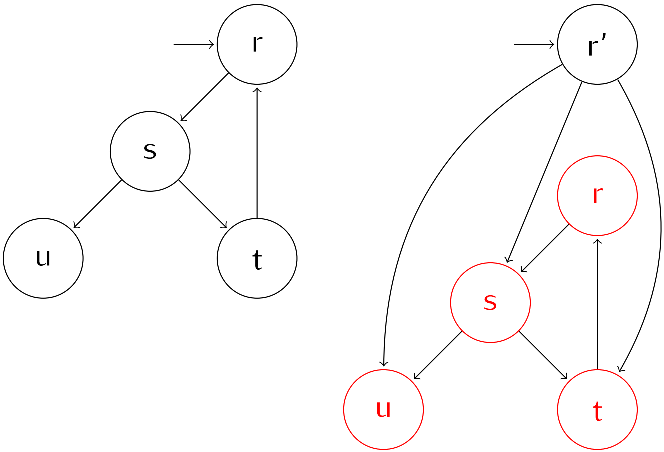
Example



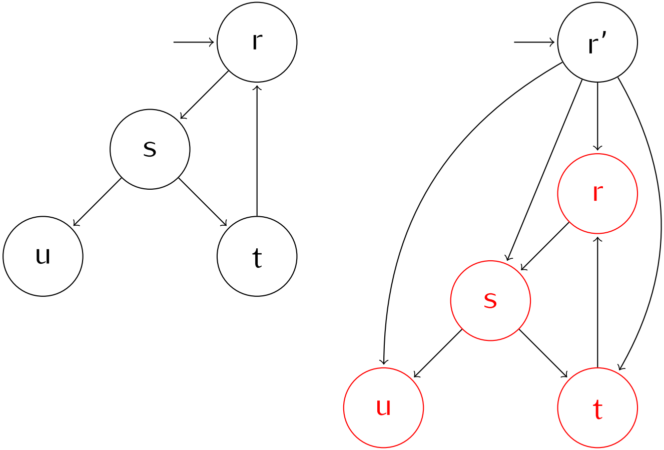
Example



Example



Example



Classical characterization of tc

We characterized the transitive closure as the set of all successors w.r.t. $\dot{\in}$.
Furthermore, we can show that tc g contains exactly those elements.

Classical characterization of tc

We characterized the transitive closure as the set of all successors w.r.t. $\dot{\in}$.
Furthermore, we can show that tc g contains exactly those elements.
The classical characterization of tc states that tc M is the least transitive superset of M, which is easy to prove from the above characterization.

Table of Contents

Basic idea

- For any type X with a relation R and suitable conversion functions between X and $\mathbb{N}$, we can construct the quotient type ${}^{X}/_{R}$

Basic idea

- For any type X with a relation R and suitable conversion functions between X and $\mathbb{N}$, we can construct the quotient type ${}^{X}/_{R}$
- Graphs have such conversion functions

Basic idea

- For any type X with a relation R and suitable conversion functions between X and $\mathbb{N}$, we can construct the quotient type ${}^{X}/_{R}$
- Graphs have such conversion functions
- Lift constructions for $\mathbb{G}$ to ${}^{\mathbb{G}}/_{\approx}$

Construction for the quotient type

Given any type X, a decidable equivalence relation R and conversion functions f : $X \to \mathbb{N}$, $f^{-1} : \mathbb{N} \to X$ such that

Construction for the quotient type

Given any type X, a decidable equivalence relation R and conversion functions f : $X \to \mathbb{N}$, $f^{-1} : \mathbb{N} \to X$ such that

- $\forall x\, y.\, Rxy \implies f\, x = f\, y$

Construction for the quotient type

Given any type X, a decidable equivalence relation R and conversion functions f : $X \to \mathbb{N}$, $f^{-1} : \mathbb{N} \to X$ such that

- $\forall x\, y.\, Rxy \implies f\, x = f\, y$
- $\forall x.\, R(f^{-1}(f\, x))x$

Construction for the quotient type

Given any type X, a decidable equivalence relation R and conversion functions f : $X \to \mathbb{N}$, $f^{-1} : \mathbb{N} \to X$ such that

- $\forall x\, y.\, Rxy \implies f\,x = f\,y$
- $\forall x.\, R(f^{-1}(f\,x))x$

we can construct the quotient type X/R as follows:

Definition

$X/R := \{n \mid f(f^{-1}\,n) = n\}$

Equivalence classes

Note that for any x, $f(f^{-1}(f\,x)) = f\,x$ holds, due to the properties of f and f^{-1}.

Equivalence classes

Note that for any x, $f(f^{-1}(f\,x)) = f\,x$ holds, due to the properties of f and f^{-1}.

Definition (equivalence classes)

norm x := (f x, A), where A is a proof that $f(f^{-1}(f\,x)) = f\,x$.

Equivalence classes

Note that for any x, $f(f^{-1}(f\,x)) = f\,x$ holds, due to the properties of f and f^{-1}.

Definition (equivalence classes)

norm x := (f x, A), where A is a proof that $f(f^{-1}(f\,x)) = f\,x$.

Definition (representative elements)

repr (n, _) := $f^{-1}\,n$

Properties of repr and norm

Equivalence classes and their representative elements work as expected, i.e.:

- $\forall x\, y.\, R\, x\, y \iff norm\ x = norm\ y$
- $\forall a\, b.\, a = b \iff R(repr\ a)(repr\ b)$

Conversion between $\mathbb{G}$ and $\mathbb{N}$

Goal: construct suitable conversion functions $\mathbb{G} \leftrightarrow \mathbb{N}$.

Conversion between $\mathbb{G}$ and $\mathbb{N}$

Goal: construct suitable conversion functions $\mathbb{G} \leftrightarrow \mathbb{N}$.

- Every graph is bisimilar to a graph over $\mathbb{N}$

Conversion between $\mathbb{G}$ and $\mathbb{N}$

Goal: construct suitable conversion functions $\mathbb{G} \leftrightarrow \mathbb{N}$.

- Every graph is bisimilar to a graph over $\mathbb{N}$
- We can construct a list of all such graphs up to any size

Conversion between $\mathbb{G}$ and $\mathbb{N}$

Goal: construct suitable conversion functions $\mathbb{G} \leftrightarrow \mathbb{N}$.

- Every graph is bisimilar to a graph over $\mathbb{N}$
- We can construct a list of all such graphs up to any size
- Use the indices of such a list to convert between $\mathbb{G}$ and $\mathbb{N}$

Well-formed grpahs

Definition (well-formed)

We call a graph well-formed if its domain starts with its root and does not contain any duplicates.

Well-formed grpahs

Definition (well-formed)

We call a graph well-formed if its domain starts with its root and does not contain any duplicates.

Lemma (Reordering lemma)

Every graph is bisimilar to a well-formed graph. We call such a graph well-formed.

Graphs over $\mathbb{N}$

Given a well-formed graph g, construct a graph $\overline{g}$ over natural numbers such that g $\approx \overline{g}$.

Graphs over $\mathbb{N}$

Given a well-formed graph g, construct a graph $\overline{g}$ over natural numbers such that g $\approx \overline{g}$.

- For every vertex v $\in$ dom g, index v $\in$ dom $\overline{g}$.

Graphs over $\mathbb{N}$

Given a well-formed graph g, construct a graph $\overline{g}$ over natural numbers such that g $\approx \overline{g}$.

- For every vertex v $\in$ dom g, index v $\in$ dom $\overline{g}$.
- For any x y, E (index x) (index y) := E x y.

Graphs over $\mathbb{N}$

Given a well-formed graph g, construct a graph $\overline{g}$ over natural numbers such that $g \approx \overline{g}$.

- For every vertex $v \in$ dom g, index $v \in$ dom $\overline{g}$.
- For any x y, E (index x) (index y) := E x y.
- We can convert back from the index to the element it corresponds to by taking the nth element of the domain from g, since g is well-formed

Graphs over $\mathbb{N}$

Given a well-formed graph g, construct a graph $\overline{g}$ over natural numbers such that $g \approx \overline{g}$.

- For every vertex $v \in \text{dom } g$, index $v \in \text{dom } \overline{g}$.
- For any x y, E (index x) (index y) := E x y.
- We can convert back from the index to the element it corresponds to by taking the nth element of the domain from g, since g is well-formed
- g and $\overline{g}$ are isomorphic on their domains, hence bisimilar.

Graphs over $\mathbb{N}$

Given a well-formed graph g, construct a graph $\overline{g}$ over natural numbers such that $g \approx \overline{g}$.

- For every vertex $v \in$ dom g, index $v \in$ dom $\overline{g}$.
- For any x y, E (index x) (index y) := E x y.
- We can convert back from the index to the element it corresponds to by taking the nth element of the domain from g, since g is well-formed
- g and $\overline{g}$ are isomorphic on their domains, hence bisimilar.
- Note that 0 is the root of $\overline{g}$.

Enumerating graphs over $\mathbb{N}$

- For any xs, ys, we can construct a list of all relations between xs and ys.

Enumerating graphs over $\mathbb{N}$

- For any xs, ys, we can construct a list of all relations between xs and ys.
- In particular, this works for xs = ys = [0, 1, ..., n-1]

Enumerating graphs over $\mathbb{N}$

- For any xs, ys, we can construct a list of all relations between xs and ys.
- In particular, this works for xs = ys = [0, 1, ..., n-1]
- For every graph g that is well-formed and has $|dom\, g| = n$, $\overline{g}$ has this form.

Enumerating graphs over $\mathbb{N}$

- For any xs, ys, we can construct a list of all relations between xs and ys.
- In particular, this works for xs = ys = [0, 1, ..., n-1]
- For every graph g that is well-formed and has $|dom\, g| = n$, $\overline{g}$ has this form.
- We can then proceed to enumerate all such graphs up to a fixed domain size.

$\mathbb{G} \leftrightarrow \mathbb{N}$

We can use these lists of graphs over $\mathbb{N}$ up to an arbitrary size to convert between graphs and natural numbers:

$\mathbb{G} \leftrightarrow \mathbb{N}$

We can use these lists of graphs over $\mathbb{N}$ up to an arbitrary size to convert between graphs and natural numbers:

- f : Given a graph g, we find the index of the first graph g' in a large enough list of graphs over $\mathbb{N}$ such that $g \approx g'$.

$\mathbb{G} \leftrightarrow \mathbb{N}$

We can use these lists of graphs over $\mathbb{N}$ up to an arbitrary size to convert between graphs and natural numbers:

- f : Given a graph g, we find the index of the first graph g' in a large enough list of graphs over $\mathbb{N}$ such that g $\approx$ g'.
- f^{-1} : Given n $\in \mathbb{N}$, we return the nth graph in a large enough list of graphs over $\mathbb{N}$

$\mathbb{G} \leftrightarrow \mathbb{N}$

We can use these lists of graphs over $\mathbb{N}$ up to an arbitrary size to convert between graphs and natural numbers:

- f : Given a graph g, we find the index of the first graph g' in a large enough list of graphs over $\mathbb{N}$ such that g $\approx$ g'.
- f^{-1} : Given n $\in \mathbb{N}$, we return the nth graph in a large enough list of graphs over $\mathbb{N}$
- $\forall g.\, g \approx f^{-1}(f\, g)$.

$\mathbb{G} \leftrightarrow \mathbb{N}$

We can use these lists of graphs over $\mathbb{N}$ up to an arbitrary size to convert between graphs and natural numbers:

- f : Given a graph g, we find the index of the first graph g' in a large enough list of graphs over $\mathbb{N}$ such that g $\approx$ g'.
- f^{-1} : Given n $\in \mathbb{N}$, we return the nth graph in a large enough list of graphs over $\mathbb{N}$
- $\forall g.\, g \approx f^{-1}(f\ g)$.
- $\forall g\ g'.\, g \approx g' \iff f\ g = f\ g'$.

$\mathcal{N}$

We know that $\mathbb{G}$ with the decidable equivalence relation $\approx$ has suitable conversion functions f, f^{-1}.
Hence, we can construct the quotient type $\mathcal{N} := \mathbb{G}/_{\approx}$.
We can lift the definitions and constructions we have for $\mathbb{G}$ to $\mathcal{N}$ by using the conversion functions norm and repr.

Table of Contents

Choice function

Choice function on graphs has to respect $\approx$. We have done that already by constructing $\mathbb{G}/_{\approx}$.

Choice function

Choice function on graphs has to respect $\approx$. We have done that already by constructing $\mathbb{G}/_{\approx}$.

Definition

$$\gamma M := \begin{cases} \emptyset & \mathit{child\_nodes}\,(\mathit{repr}\ M) = [] \\ \mathit{norm}\,(\mathit{subgraph}\ x) & \mathit{child\_nodes}\,(\mathit{repr}\ M) = x :: xs \end{cases}$$

Choice function

Choice function on graphs has to respect $\approx$. We have done that already by constructing $\mathbb{G}/_{\approx}$.

Definition

$$\gamma M := \begin{cases} \emptyset & \mathit{child\_nodes}\,(\mathit{repr}\ M) = [] \\ \mathit{norm}\,(\mathit{subgraph}\ x) & \mathit{child\_nodes}\,(\mathit{repr}\ M) = x :: xs \end{cases}$$

It is easy to see that $\forall M \neq \emptyset.\ \gamma M \in M$.

Future Work

- Enumerability of $\mathbb{T}$ using Ackermann's encoding
- Quotient type $\mathbb{T}/_{\equiv}$
- ZF(C) constructions for $\mathbb{T}$
- Relation between CCS with recursion and $\mathcal{N}$

Bibliography I

- Samson Abramsky, *A Cook's Tour of the Finitary Non-Well-Founded Sets*, CoRR **abs/1111.7148** (2011).
- W. Ackermann, *Die Widerspruchsfreiheit der allgemeinen Mengenlehre*, Mathematische Annalen **114** (1937), 305–315.
- Peter Aczel, *Non-well-founded sets*, Lecture Notes, no. 14, Center for the Study of Language and Information, Stanford University, 1988.
- Michael Baldamus, *A Non-well-founded Sets Semantics for Observation Congruence over Full CCS*, Tech. report, 1994.

Bibliography II

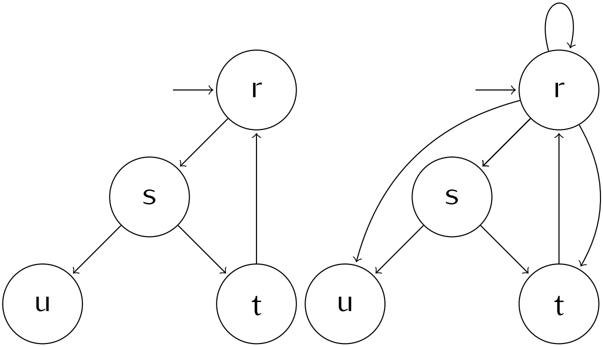
- Chad Brown, *Ackermann Encoding of Decidable Hereditarily Finite Sets*, `https://www.ps.uni-saarland.de/settheory/HF/HF.v`, [Online; accessed 19-July-2015].
- Dominik Kirst, *Formalised Set Theory: Well-Orderings and the Axiom of Choice*, Bachelor's thesis, Saarland University, August 2014.
- R. Milner, *A calculus of communicating systems*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

Bibliography III

- Alexandre Miquel, *Inconsistent Type Systems*, `http://www.cse.chalmers.se/research/group/logic/TypesSS05/Extra/miquel_sl3.pdf`, August 2005, [Online; accessed 19-July-2015].
- Kathrin Stark, *Quantitative Recursion-Free Process Axiomatization in Coq*, Bachelor's thesis, Saarland University, May 2014.
- The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*, `http://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

Faulty construction for tc

Not adding a new root, but changing the edges going out from the old root does not work:



On the right hand side, the original graph structure is lost!

Children and Subgraphs Definition

We can use E to define the children of a graph, which are basically its elements (modulo bisimulation).

Definition

child_nodes g $:=$ *filter* ($\lambda x.\, E(rootg)x = true$) (*dom g*)
subgraph (*x* : *g*) $:=$ *G* (*edgeRel g*) (*dom g*) *x*
children g $:=$ *map subgraph* (*child_nodes g*)

$\dot{\in}^n$

The notion of successors w.r.t. $\dot{\in}$ can be captured inductively:

Definition ($\dot{\in}^n$)

$$\frac{g_1 \approx g_2}{g_1 \mathrel{\dot{\in}^0} g_2}$$

$$\frac{g_1 \mathrel{\dot{\in}} g_2 \qquad g_2 \mathrel{\dot{\in}^n} g_3}{g_1 \mathrel{\dot{\in}^{Sn}} g_3}$$

Correspondence between $\rightarrow^*$ and $\dot{\in}^n$

- $r \rightarrow^* x \implies g_1 \approx \text{subgraph x} \implies$
  $\exists n.\, g1 \,\dot{\in}^n\, \text{(sugraph r)}.$
- $g_1 \,\dot{\in}^n g_2 \implies \mathit{root}\, g_2 \in \mathit{dom}\, g_2 \implies$
  $\exists x.\, \mathit{root}\, g2 \rightarrow^* x \wedge g_1 \approx \text{subgraph x}$
- $r \rightarrow^+ x \implies g_1 \approx \text{subgraph x} \implies$
  $\exists n > 0.\, g1 \,\dot{\in}^n (\mathit{sugraph}\, r).$
- $n > 0 \implies g1 \,\dot{\in}^n g2 \implies$
  $\exists x.(\mathit{root}\, g2) \rightarrow^+ x \wedge g_1 \approx \text{subgraph x}.$

Transitive closure

Definition (transitive closure)

tc g := *G f (None :: (map Some (dom g))) (None) where*
f (Some x) (Some y) := *edgeRel x y*
f None (Some y) := $(\mathit{root}\, g) \rightarrow^{+} y$.
f _ _ := *false*

Due to the correspondence between $\rightarrow^{n}$ and $\dot{\in}^{n}$, it is easy to see that
$\forall g\, g'.\, g' \dot{\in} (tc\, g) \iff \exists n > 0.\, g' \dot{\in}^{n} g$.

proof tc transitive

Definition (transitive graph)

g_1 *is transitive* $:= \forall g_2\, g_3 .\, g_3 \mathrel{\dot{\in}} g_2 \implies g_2 \mathrel{\dot{\in}} g_1 \implies g_3 \mathrel{\dot{\in}} g_1 .$

Lemma

tc g is transitive.

proof tc transitive

Definition (transitive graph)

g_1 is transitive $:= \forall g_2\, g_3\,.\, g_3 \dot{\in} g_2 \implies g_2 \dot{\in} g_1 \implies g_3 \dot{\in} g_1\,.$

Lemma

tc g is transitive.

Let g” $\dot{\in}$ g’ $\dot{\in}$ tc g.
We know that g’ $\dot{\in}^{n}$ g for some $n > 0$.
Hence, g” $\dot{\in}^{S\,n}$ g, which in turn implies g” $\dot{\in}$ tc g.

proof tc superset

Lemma

$g \mathrel{\dot{\subseteq}} tc\ g.$

Let g′ $\dot{\in}$ g, i.e. there is
x : g such that E (root g) x = true and g′ ≈ subgraph g.
Note that root g $\rightarrow^{+}$ x, hence E None (Some x) = true.
g′ $\dot{\in}$ tc g follows from the fact that subgraph x ≈ subgraph (Some x).

proof tc least such set

Lemma

$\forall g^*.\, \mathit{transitive}\, g^* \implies g \,\dot{\subseteq}\, g^* \implies \mathit{tc}\, g \,\dot{\subseteq}\, g^*$

Let g^* be transitive and $g \,\dot{\subseteq}\, g^*$.

proof tc least such set

Lemma

$\forall g^*.\ \mathit{transitive}\ g^* \implies g \,\dot{\subseteq}\, g^* \implies \mathit{tc}\ g \,\dot{\subseteq}\, g^*$

Let g^* be transitive and $g \,\dot{\subseteq}\, g^*$.
We show $\forall n > 0 \forall g'.\ g' \,\dot{\in}\,^n g \implies g' \,\dot{\in}\, g^*$
by induction on n. The base case is trivial.
In the inductive case, n = S n' and $g' \,\dot{\in}\,^{S\,n'} g$.

proof tc least such set

Lemma

$\forall g^*.\, \mathit{transitive}\, g^* \implies g \mathrel{\dot{\subseteq}} g^* \implies \mathit{tc}\, g \mathrel{\dot{\subseteq}} g^*$

Let g^* be transitive and $g \mathrel{\dot{\subseteq}} g^*$.
We show $\forall n > 0 \forall g'.\, g' \mathrel{\dot{\in}}^{n} g \implies g' \mathrel{\dot{\in}} g^*$
by induction on n. The base case is trivial.
In the inductive case, n = S n' and $g' \mathrel{\dot{\in}}^{S\, n'} g$.

proof tc least such set

Lemma

$\forall g^*.\, transitive\, g^* \implies g \dot{\subseteq} g^* \implies tc\, g \dot{\subseteq} g^*$

Let g^* be transitive and $g \dot{\subseteq} g^*$.
We show $\forall n > 0 \forall g'.\, g' \dot{\in}^n g \implies g' \dot{\in} g^*$
by induction on n. The base case is trivial.
In the inductive case, n = S n' and $g' \dot{\in}^{S\, n'} g$.

- If n' = 0, i.e. n = 1, $g' \dot{\in}^1 g \iff g' \dot{\in} g$, which immediately gives us that $g' \dot{\in} g^*$, since $g \dot{\subseteq} g^*$.

proof tc least such set

Lemma

$\forall g^{*}.\, \mathit{transitive}\, g^{*} \implies g \mathrel{\dot{\subseteq}} g^{*} \implies \mathit{tc}\, g \mathrel{\dot{\subseteq}} g^{*}$

Let g^{*} be transitive and $g \mathrel{\dot{\subseteq}} g^{*}$.
We show $\forall n > 0 \forall g'.\, g' \mathrel{\dot{\in}}{}^{n} g \implies g' \mathrel{\dot{\in}} g^{*}$
by induction on n. The base case is trivial.
In the inductive case, n = S n' and $g' \mathrel{\dot{\in}}{}^{S\, n'} g$.

- If n' = 0, i.e. n = 1, $g' \mathrel{\dot{\in}}{}^{1} g \iff g' \mathrel{\dot{\in}} g$, which immediately gives us that $g' \mathrel{\dot{\in}} g^{*}$, since $g \mathrel{\dot{\subseteq}} g^{*}$.
- Otherwise, n' = S m. Since $g' \mathrel{\dot{\in}}{}^{S\, n'} g$, there is some graph h such that $g' \mathrel{\dot{\in}} h$ and $h \mathrel{\dot{\in}}{}^{S\, m} g$.

proof tc least such set

Lemma

$\forall g^*.\, \mathit{transitive}\, g^* \implies g \,\dot{\subseteq}\, g^* \implies tc\, g \,\dot{\subseteq}\, g^*$

Let g^* be transitive and $g \,\dot{\subseteq}\, g^*$.
We show $\forall n > 0 \forall g'.\, g' \,\dot{\in}^{n} g \implies g' \,\dot{\in}\, g^*$
by induction on n. The base case is trivial.
In the inductive case, n = S n' and $g' \,\dot{\in}^{S\, n'} g$.

- If n' = 0, i.e. n = 1, $g' \,\dot{\in}^{1} g \iff g' \,\dot{\in}\, g$, which immediately gives us that $g' \,\dot{\in}\, g^*$, since $g \,\dot{\subseteq}\, g^*$.
- Otherwise, n' = S m. Since $g' \,\dot{\in}^{S\, n'} g$, there is some graph h such that $g' \,\dot{\in}\, h$ and $h \,\dot{\in}^{S\, m} g$.
- By IH, we know that $h \,\dot{\in}\, g^*$, hence $h \,\dot{\subseteq}\, g^*$.

Reordering lemma proof

Lemma (Reordering lemma)

Every graph is bisimilar to a well-formed graph.

- If dom g = [], we know that g $\approx \dot{\emptyset}$.
  (Recall that $\dot{\emptyset}$:= G true [tt] tt, hence we now have one more element in the domain).
- Otherwise, reorder as follows:
  dom g′ = root (dom g) :: rem (root g) (undup dom g)
  (In this step, the domain size can only decrease, not increase)

This gives us even more: Every graph is bisimilar to a well-formed graph whose domain contains at most one more vertex.

Transition Functions

We know that
allFuns xs ys := $map(\lambda A.\lambda x\,y.(x,y) \in A)(\mathcal{P}(xs \times ys))$
contains all possible relations on xs and ys.
We can use this to construct all transition functions for a given graph.

Definition

range n := [0, 1, ..., n-1]

Note that for any well-formed graph g with |dom g| = n, dom $\overline{g}$ = range n.

Enumerating graphs

Definition

$\alpha\, n := map\ (\lambda f\,.\,G\ f\ (range\ n)\ 0)$
$(allFuns\ (range\ n)\ (range\ n))$

$\alpha\, n$ yields a list of all graphs over natural numbers with domain range n and 0 as the root.

Definition

$\beta\, n := mapcat\ \alpha\ [1..n]$

$\beta\, n$ contains a list of all such graphs whose domain has size at most n.

Properties of α and β

There are a few important properties of α and β:

- Note that $\forall g. \overline{g} \in \beta(S|dom\, g|)$.
- Likewise, since $|\alpha n| >= 1$ for any n, $|beta\, n| >= n$.
- This means that for any n, $\beta(Sn)$ supports indices from 0 to at least n.

$\mathbb{G} \leftrightarrow \mathbb{N}$

Conversion functions with exact numbers:

- f : Given a graph g, we find the index of the first graph g' in $\beta(S\,|dom\,g|)$ such that g $\approx$ g'.
- f^{-1} : Given n $\in \mathbb{N}$, we return the nth graph in $\beta(S\,n)$
- $\forall g.\, g \approx f^{-1}(f\,g)$.
- $\forall g\, g'.\, g \approx g' \iff f\,g = f\,g'$.

Properties of repr and norm

Note the following important properties of repr and norm:

- $\forall x\, y.\, R\, x\, y \iff repr\,(norm\, x) = repr\,(norm\, y)$
- $\forall x\, y. repr\,(norm\, x) = repr\,(norm\, y) \iff norm\, x = norm\, y$

Second point rather technical:

Lemma eq_dep_dec_sig (x y : X) (h : P x)
(h' : P y) (p : x = y)
(q : match p with eq_refl ⇒ h end = h') :
exist P x h = exist P y h'.
Proof. now destruct p,q. Qed.