Quotients of Bounded Natural Functors

Basil Fürer

Andreas Lochbihler

Digital Asset

ETH zürich

Joshua Schneider

Dmitriy Traytel

Dramatis personae

Andreas

Isabelle Expert

Dmitriy

Working Formalizer
and Narrator

Isabelle

Proof Assistant

The characters and incidents portrayed and the names used herein are fictitious and any resemblance to the names, character, or history of any person is coincidental and unintentional.

A formalization problem

datatype a *re* $=$ Atom a | Alt (a *re*) (a *re*) | Conc (a *re*) (a *re*) | Star (a *re*)

A formalization problem

datatype a *re* $=$ Atom a | Alt (a *re*) (a *re*) | Conc (a *re*) (a *re*) | Star (a *re*)

datatype *ldl* $=$ Prop *string* | And *ldl ldl* | Neg *ldl* | Match (*ldl re*)

A formalization problem

datatype a *re* $=$ Atom a | Alt (a *re*) (a *re*) | Conc (a *re*) (a *re*) | Star (a *re*)

inductive $\sim_{ACI}$ **where**

$$\text{Alt}\ (\text{Alt}\ r\ s)\ t \sim_{ACI} \text{Alt}\ r\ (\text{Alt}\ s\ t) \qquad \text{Alt}\ r\ s \sim_{ACI} \text{Alt}\ s\ r \qquad \text{Alt}\ r\ r \sim_{ACI} r$$

$$\frac{r \sim_{ACI} r' \quad s \sim_{ACI} s'}{\text{Alt}\ r\ s \sim_{ACI} \text{Alt}\ r'\ s'} \qquad \frac{r \sim_{ACI} r' \quad s \sim_{ACI} s'}{\text{Conc}\ r\ s \sim_{ACI} \text{Conc}\ r'\ s'} \qquad \frac{r \sim_{ACI} r'}{\text{Star}\ r \sim_{ACI} \text{Star}\ r'}$$

$$r \sim_{ACI} r \qquad \frac{r \sim_{ACI} s}{s \sim_{ACI} r} \qquad \frac{r \sim_{ACI} s \quad s \sim_{ACI} t}{r \sim_{ACI} t}$$

datatype *ldl* $=$ Prop *string* | And *ldl ldl* | Neg *ldl* | Match (*ldl re*)

A formalization problem

datatype a *re* $=$ Atom a | Alt (a *re*) (a *re*) | Conc (a *re*) (a *re*) | Star (a *re*)

inductive $\sim_{ACI}$ **where**

$$\text{Alt}\ (\text{Alt}\ r\ s)\ t \sim_{ACI} \text{Alt}\ r\ (\text{Alt}\ s\ t) \qquad \text{Alt}\ r\ s \sim_{ACI} \text{Alt}\ s\ r \qquad \text{Alt}\ r\ r \sim_{ACI} r$$

$$\frac{r \sim_{ACI} r' \quad s \sim_{ACI} s'}{\text{Alt}\ r\ s \sim_{ACI} \text{Alt}\ r'\ s'} \qquad \frac{r \sim_{ACI} r' \quad s \sim_{ACI} s'}{\text{Conc}\ r\ s \sim_{ACI} \text{Conc}\ r'\ s'} \qquad \frac{r \sim_{ACI} r'}{\text{Star}\ r \sim_{ACI} \text{Star}\ r'}$$

$$r \sim_{ACI} r \qquad \frac{r \sim_{ACI} s}{s \sim_{ACI} r} \qquad \frac{r \sim_{ACI} s \quad s \sim_{ACI} t}{r \sim_{ACI} t}$$

quotient_type a $re_{ACI} = a\ re / \sim_{ACI}$

datatype *ldl* $=$ Prop *string* | And *ldl ldl* | Neg *ldl* | Match (*ldl* re_{ACI})

A formalization problem

datatype *a re* = Atom *a* | Alt (*a re*) (*a re*) | Conc (*a re*) (*a re*) | Star (*a re*)

inductive $\sim_{ACI}$ **where**

A ... r s $\sim_{ACI}$ Alt s r   Alt r r $\sim_{ACI}$ r

$$\frac{\ldots_{ACI}\ r' \quad s \sim_{ACI} s'}{\ldots s \sim_{ACI} \text{Conc } r'\ s'} \qquad \frac{r \sim_{ACI} r'}{\text{Star } r \sim_{ACI} \text{Star } r'}$$

$$\frac{r \sim_{ACI} s}{s \sim_{ACI} r} \qquad \frac{r \sim_{ACI} s \quad s \sim_{ACI} t}{r \sim_{ACI} t}$$

Unsupported recursive occurrence of type *ldl* via type constructor re_{ACI} in type expression *ldl* re_{ACI}.

Use the **bnf** command to register re_{ACI} as a bounded natural functor to allow nested (co)recursion through it.

qu... _type ... $_{ACI}$ ... $_{ACI}$

datatype *ldl* = Prop *string* | And *ldl ldl* | Neg *ldl* | Match (*ldl* re_{ACI})

Interlude: Contribution

Identified sufficient conditions on when quotients of BNFs are BNFs

Relevant for (co)datatypes, relational parametricity, refinement

Interlude: Contribution

Identified sufficient conditions on when quotients of BNFs are BNFs

Relevant for (co)datatypes, relational parametricity, refinement

Automated BNF preservation proofs via **lift_bnf** command in Isabelle HOL

Datatype recursion worries

Higher Order Logic Theorem Proving and its Applications (A-20)
L.J.M. Claesen and M.J.C. Gordon (Editors)
Elsevier Science Publishers B.V. (North-Holland) 561

Why We Can't have SML Style datatype Declarations in HOL

Elsa L. Gunter

AT&T Bell Laboratories, Rm. #2A-432, Murray Hill, NJ, 07974-0636, USA

Unsupported recursive occurrence of type *ldl* via type constructor re_{ACI} in type expression *ldl* re_{ACI}.

Use the **bnf** command to register re_{ACI} as a bounded natural functor to allow nested (co)recursion through it.

Datatype recursion worries

Higher Order Logic Theorem Proving and its Applications (A-20)
L.J.M. Claesen and M.J.C. Gordon (Editors)
Elsevier Science Publishers B.V. (North-Holland) 561

Why We Can't have SML Style datatype Declarations in HOL

Elsa L. Gunter

AT&T Bell Laboratories, Rm. #2A-432, Murray Hill, NJ, 07974-0636, USA

```
Unsupported recursive occurrence
of type Idl via type constructor
re_ACI in type expression Idl re_ACI.

Use the bnf command to register
re_ACI as a bounded natural functor
to allow nested (co)recursion
through it.
```

datatype *bad* = C (*bad set*) | …

C :: *bad set* ⇒ *bad* **injective**

Datatype recursion worries

Higher Order Logic Theorem Proving and its Applications (A-20)
L.J.M. Claesen and M.J.C. Gordon (Editors)
Elsevier Science Publishers B.V. (North-Holland) 561

Why We Can't have SML Style datatype Declarations in HOL

Elsa L. Gunter

AT&T Bell Laboratories, Rm. #2A-432, Murray Hill, NJ, 07974-0636, USA

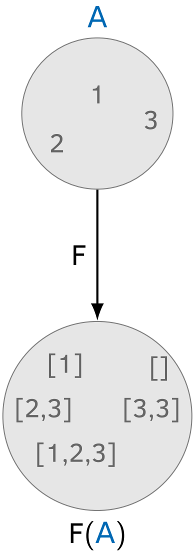
```
Unsupported recursive occurrence of type Idl via type constructor re_ACI in type expression Idl re_ACI.

Use the bnf command to register re_ACI as a bounded natural functor to allow nested (co)recursion through it.
```

datatype *bad* = C (*bad set*) | . . .

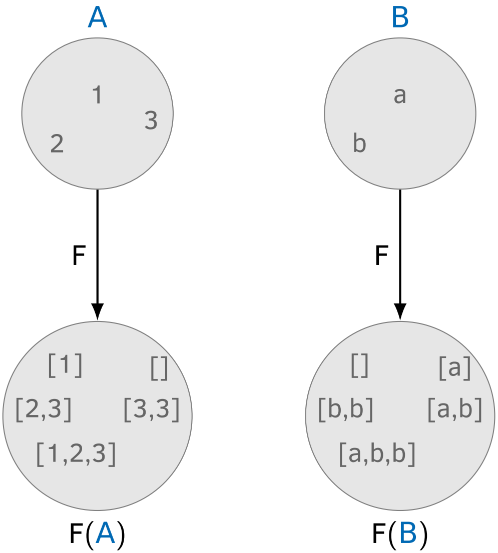
C :: *bad set* ⇒ *bad* **injective**

Datatypes may recurse only through BNFs

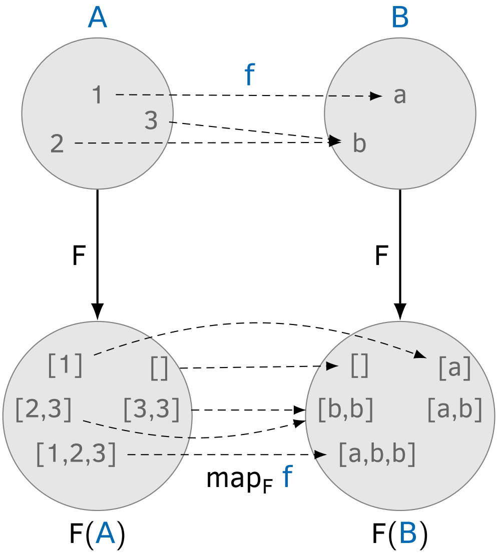
Bounded Natural Functors (BNF)



Bounded Natural Functors (BNF)



Bounded Natural Functors (BNF)

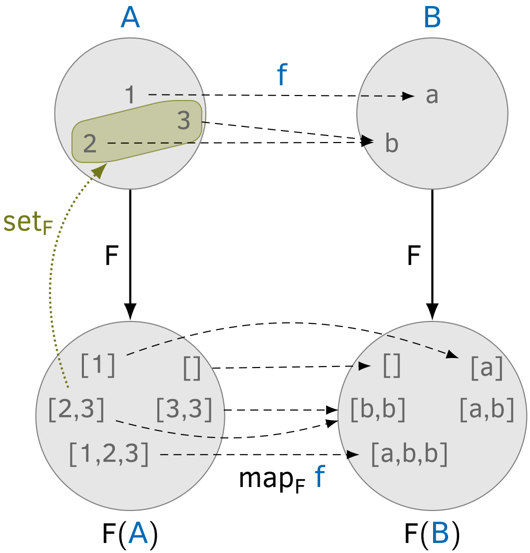


Functor

$\mathrm{map}_F\ \mathrm{id} = \mathrm{id}$

$\mathrm{map}_F\ g \circ \mathrm{map}_F\ f = \mathrm{map}_F\ (g \circ f)$

Bounded Natural Functors (BNF)



Functor

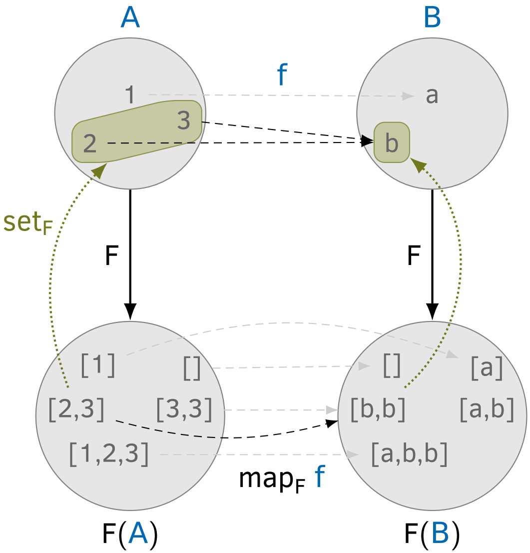
$\mathrm{map}_F\ \mathrm{id} = \mathrm{id}$

$\mathrm{map}_F\ g \circ \mathrm{map}_F\ f = \mathrm{map}_F\ (g \circ f)$

Bound

$|\mathrm{set}_F\ x| < \aleph$

Bounded Natural Functors (BNF)



Functor

$\mathrm{map}_F\ \mathrm{id} = \mathrm{id}$

$\mathrm{map}_F\ g \circ \mathrm{map}_F\ f = \mathrm{map}_F\ (g \circ f)$

Bound

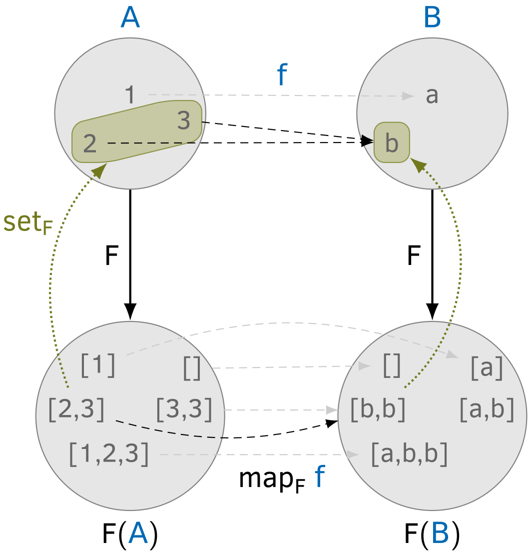
$|\mathrm{set}_F\ x| < \aleph$

Natural

$\mathrm{set}_F\ (\mathrm{map}_F\ f\ x) = f\langle \mathrm{set}_F\ x \rangle$

$$\frac{\forall x \in \mathrm{set}_F\ x.\ f\ x = g\ x}{\mathrm{map}_F\ f\ x = \mathrm{map}_F\ g\ x}$$

Bounded Natural Functors (BNF)



Functor

$\mathsf{map}_F\ \mathsf{id} = \mathsf{id}$

$\mathsf{map}_F\ g \circ \mathsf{map}_F\ f = \mathsf{map}_F\ (g \circ f)$

Bound

$|\mathsf{set}_F\ x| < \aleph$

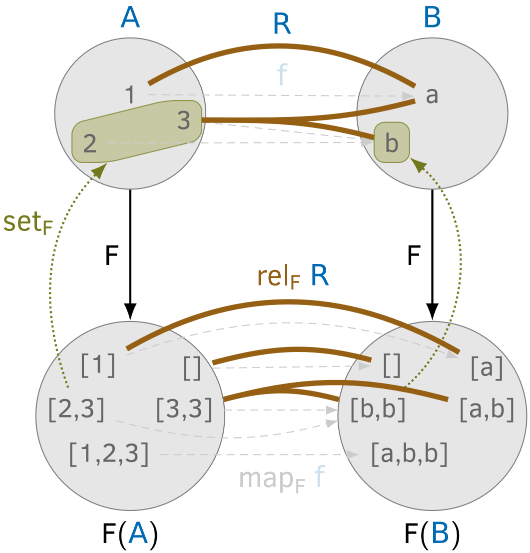
Natural

$\mathsf{set}_F\ (\mathsf{map}_F\ f\ x) = f\langle \mathsf{set}_F\ x \rangle$

$\boxed{F(\bigcap \mathcal{A}) = \bigcap F\langle \mathcal{A} \rangle}$

$$\frac{\forall x \in \mathsf{set}_F\ x.\ f\ x = g\ x}{\mathsf{map}_F\ f\ x = \mathsf{map}_F\ g\ x}$$

Bounded Natural Functors (BNF)



Functor

$$\mathsf{map}_F\ \mathsf{id} = \mathsf{id}$$

$$\mathsf{map}_F\ g \circ \mathsf{map}_F\ f = \mathsf{map}_F\ (g \circ f)$$

Bound

$$|\mathsf{set}_F\ x| < \aleph$$

$$\boxed{F(\bigcap \mathcal{A}) = \bigcap F\langle \mathcal{A} \rangle}$$

Natural

$$\mathsf{set}_F\ (\mathsf{map}_F\ f\ x) = f\langle \mathsf{set}_F\ x \rangle$$

$$\frac{\forall x \in \mathsf{set}_F\ x.\ f\ x = g\ x}{\mathsf{map}_F\ f\ x = \mathsf{map}_F\ g\ x}$$

Relator

$$(x, y) \in \mathsf{rel}_F\ R = \exists z \in F(R).\ \mathsf{map}_F\ \pi_1\ z = x\ \wedge \\ \mathsf{map}_F\ \pi_2\ z = y$$

$$\mathsf{rel}_F\ R \bullet \mathsf{rel}_F\ S = \mathsf{rel}_F\ (R \bullet S)$$

Closure properties of BNF

Basic BNFs

_ + _ _ × _

$\tau \Rightarrow$ _

_ set _ $\Rightarrow \tau$

Non-BNFs

Closure properties of BNF

Basic BNFs

_ + _   _ × _

$\tau \Rightarrow$ _

Derived BNFs

composition

unit + _ × _

codatatypes

_ stream

datatypes

_ list

subtypes*

_ balanced-tree

_ set   _ $\Rightarrow \tau$

Non-BNFs

* Conditions apply.

Viewing re_{ACI} as a subtype

$\textbf{fun}\ \mathrm{nf}_{\mathrm{ACI}} :: \mathit{a}\ \mathit{re} \Rightarrow \mathit{a}\ \mathit{re}\ \textbf{where}\ \ldots$

$\textbf{lemma}\ r \sim_{\mathrm{ACI}} s \longleftrightarrow \mathrm{nf}_{\mathrm{ACI}}\ r = \mathrm{nf}_{\mathrm{ACI}}\ s \quad \langle \mathit{proof} \rangle$

$\textbf{typedef}\ \mathit{a}\ \mathit{re}_{\mathrm{ACI}} = \underbrace{\{\mathrm{nf}_{\mathrm{ACI}}\ r \mid r :: \mathit{a}\ \mathit{re}\}}_{\mathrm{NF}}\ \textbf{by}\ \mathit{auto}$

Viewing re_{ACI} as a subtype

fun $\mathrm{nf}_{\mathrm{ACI}} :: a\ re \Rightarrow a\ re$ **where** $\ldots$

lemma $r \sim_{\mathrm{ACI}} s \longleftrightarrow \mathrm{nf}_{\mathrm{ACI}}\ r = \mathrm{nf}_{\mathrm{ACI}}\ s$ $\langle proof \rangle$

typedef $a\ re_{\mathrm{ACI}} = \underbrace{\{\mathrm{nf}_{\mathrm{ACI}}\ r \mid r :: a\ re\}}_{\mathrm{NF}}$ **by** *auto*

lift_bnf $a\ re_{\mathrm{ACI}}$

1. $s \in \mathrm{NF} \longrightarrow \mathrm{map}_{re}\ f\ s \in \mathrm{NF}$
2. $\ldots$

Viewing re_{ACI} as a subtype

fun $\mathrm{nf}_{\mathrm{ACI}} :: a\ re \Rightarrow a\ re$ **where** $\ldots$

lemma $r \sim_{\mathrm{ACI}} s \longleftrightarrow \mathrm{nf}_{\mathrm{ACI}}\ r = \mathrm{nf}_{\mathrm{ACI}}\ s \quad \langle \mathit{proof} \rangle$

typedef $a\ re_{\mathrm{ACI}} = \underbrace{\{\mathrm{nf}_{\mathrm{ACI}}\ r \mid r :: a\ re\}}_{\mathrm{NF}}$ **by** *auto*

lift_bnf $a\ re_{\mathrm{ACI}}$

unlikely for non-injective f

1. $s \in \mathrm{NF} \longrightarrow \mathrm{map}_{re}\ f\ s \in \mathrm{NF}$
2. $\ldots$

Viewing re_{ACI} as a subtype

fun $\mathrm{nf}_{\mathrm{ACI}} :: \alpha\ re \Rightarrow \alpha\ re$ **where** $\ldots$

lemma $r \sim_{\mathrm{ACI}} s \longleftrightarrow \mathrm{nf}_{\mathrm{ACI}}\ r = \mathrm{nf}_{\mathrm{ACI}}\ s \quad \langle proof \rangle$

typedef $\alpha\ re_{\mathrm{ACI}} = \underbrace{\{\mathrm{nf}_{\mathrm{ACI}}\ r \mid r :: \alpha\ re\}}_{\mathrm{NF}}$ **by** *auto*

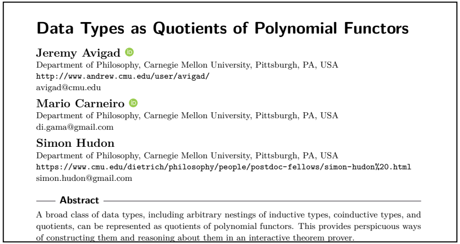
lift_bnf $\alpha\ re_{\mathrm{ACI}}$

unlikely for non-injective f

1. $s \in \mathrm{NF} \longrightarrow \mathrm{map}_{re}\ f\ s \in \mathrm{NF}$
2. $\ldots$

Quotients can be viewed as subtypes via representatives but we cannot lift the BNF structure along this view.

Quotients of Polynomial Functors


Data Types as Quotients of Polynomial Functors

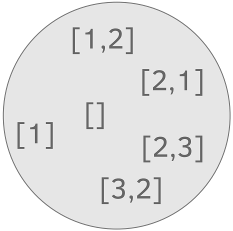
Jeremy Avigad
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.andrew.cmu.edu/user/avigad/
avigad@cmu.edu

Mario Carneiro
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
di.gama@gmail.com

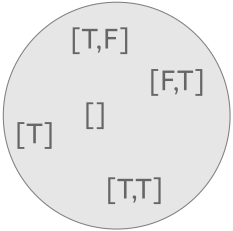
Simon Hudon
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
https://www.cmu.edu/dietrich/philosophy/people/postdoc-fellows/simon-hudon%20.html
simon.hudon@gmail.com

Abstract

A broad class of data types, including arbitrary nestings of inductive types, coinductive types, and quotients, can be represented as quotients of polynomial functors. This provides perspicuous ways of constructing them and reasoning about them in an interactive theorem prover.



F(A)



F(B)

Quotients of Polynomial Functors

Data Types as Quotients of Polynomial Functors

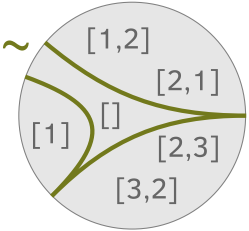
Jeremy Avigad
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.andrew.cmu.edu/user/avigad/
avigad@cmu.edu

Mario Carneiro
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
di.gama@gmail.com

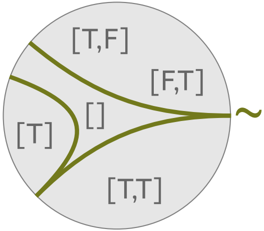
Simon Hudon
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
https://www.cmu.edu/dietrich/philosophy/people/postdoc-fellows/simon-hudon%20.html
simon.hudon@gmail.com

Abstract

A broad class of data types, including arbitrary nestings of inductive types, coinductive types, and quotients, can be represented as quotients of polynomial functors. This provides perspicuous ways of constructing them and reasoning about them in an interactive theorem prover.



F(A)/~



F(B)/~

Quotients of Polynomial Functors

Data Types as Quotients of Polynomial Functors

Jeremy Avigad
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.andrew.cmu.edu/user/avigad/
avigad@cmu.edu

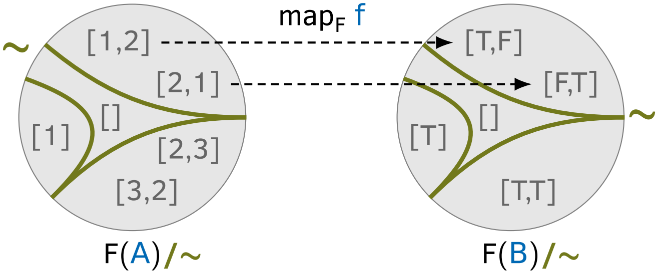
Mario Carneiro
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
di.gama@gmail.com

Simon Hudon
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
https://www.cmu.edu/dietrich/philosophy/people/postdoc-fellows/simon-hudon%20.html
simon.hudon@gmail.com

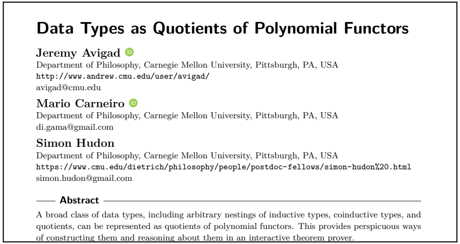
Abstract

A broad class of data types, including arbitrary nestings of inductive types, coinductive types, and quotients, can be represented as quotients of polynomial functors. This provides perspicuous ways of constructing them and reasoning about them in an interactive theorem prover.

$$x \sim y \longrightarrow \mathrm{map}_F\ f\ x \sim \mathrm{map}_F\ f\ y$$



Quotients of Polynomial Functors



Data Types as Quotients of Polynomial Functors

Jeremy Avigad
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.andrew.cmu.edu/user/avigad/
avigad@cmu.edu

Mario Carneiro
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
di.gama@gmail.com

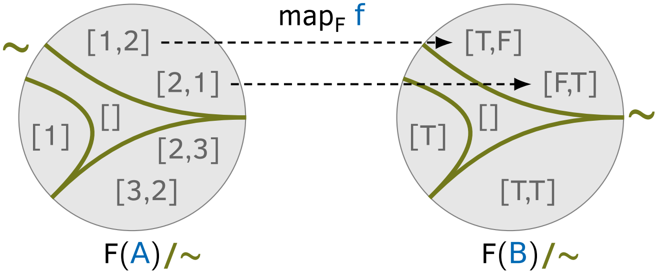
Simon Hudon
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
https://www.cmu.edu/dietrich/philosophy/people/postdoc-fellows/simon-hudon%20.html
simon.hudon@gmail.com

Abstract

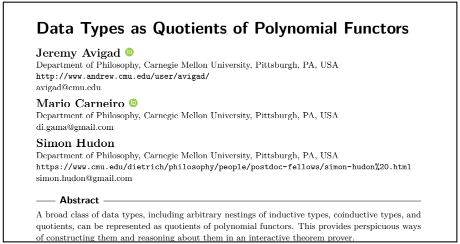
A broad class of data types, including arbitrary nestings of inductive types, coinductive types, and quotients, can be represented as quotients of polynomial functors. This provides perspicuous ways of constructing them and reasoning about them in an interactive theorem prover.

$$x \sim y \longrightarrow \mathrm{map}_F\, f\, x \sim \mathrm{map}_F\, f\, y$$

$$x \sim y \longrightarrow \mathrm{set}_F\, x = \mathrm{set}_F\, y$$



Quotients of Polynomial Functors



Data Types as Quotients of Polynomial Functors

Jeremy Avigad
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.andrew.cmu.edu/user/avigad/
avigad@cmu.edu

Mario Carneiro
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
di.gama@gmail.com

Simon Hudon
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
https://www.cmu.edu/dietrich/philosophy/people/postdoc-fellows/simon-hudon%20.html
simon.hudon@gmail.com

Abstract

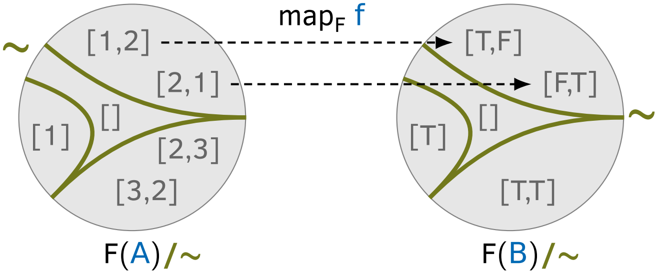
A broad class of data types, including arbitrary nestings of inductive types, coinductive types, and quotients, can be represented as quotients of polynomial functors. This provides perspicuous ways of constructing them and reasoning about them in an interactive theorem prover.

$x \sim y \longrightarrow \mathrm{map}_F\ f\ x \sim \mathrm{map}_F\ f\ y$

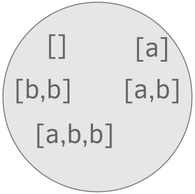
$x \sim y \longrightarrow \mathrm{set}_F\ x = \mathrm{set}_F\ y$

$\sim$ preserves wide intersections

$\sim$ preserves weak pullbacks

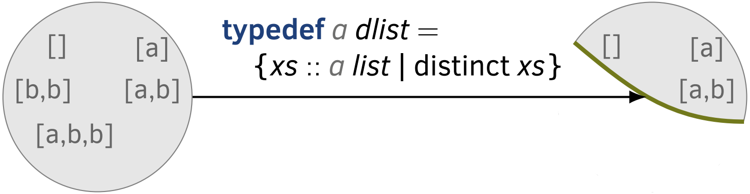


Distinct Lists

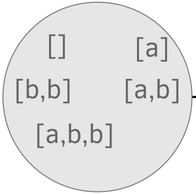


Distinct Lists

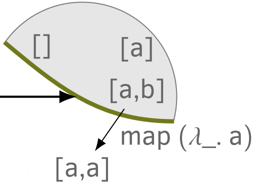
typedef *a dlist* $=$
$\{xs :: a\ list \mid \text{distinct}\ xs\}$



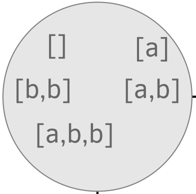
Distinct Lists



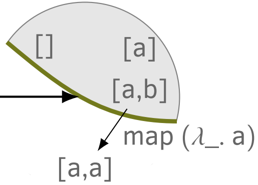
typedef a *dlist* $=$
$\{xs :: \mathit{a}\ \mathit{list} \mid \text{distinct}\ xs\}$



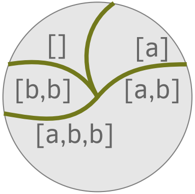
Distinct Lists



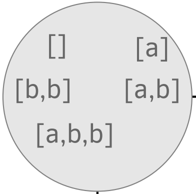
typedef α *dlist* $=$
$\{xs :: \alpha\ list \mid \text{distinct}\ xs\}$



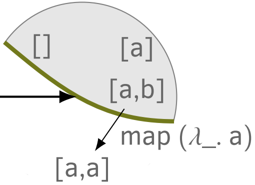
quotient_type α *dlist* $=$
$\alpha\ list\ /\ \underbrace{(\lambda xs\ ys.\ \text{remdups}\ xs = \text{remdups}\ ys)}_{\sim_{\text{dlist}}}$



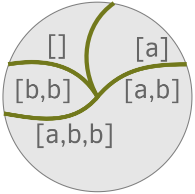
Distinct Lists



typedef *a dlist* =
{*xs* :: *a list* | distinct *xs*}



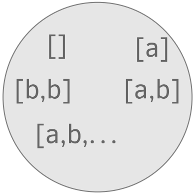
quotient_type *a dlist* =
$a\ list\ /\ \underbrace{(\lambda xs\ ys.\ \mathrm{remdups}\ xs = \mathrm{remdups}\ ys)}_{\sim_{\mathrm{dlist}}}$



- ✓ $xs \sim_{\mathrm{dlist}} ys \longrightarrow \mathrm{set}\ xs = \mathrm{set}\ ys$
- ✓ $\sim_{\mathrm{dlist}}$ preserves wide intersections
- ✓ $\sim_{\mathrm{dlist}}$ preserves weak pullbacks

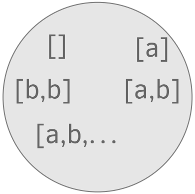
Terminated Lazy Lists

codatatype *a llist* = LNil | LCons *a* (*a llist*)

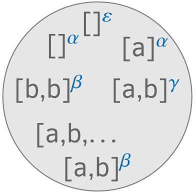


Terminated Lazy Lists

codatatype *a* *llist* = LNil | LCons *a* (*a* *llist*)

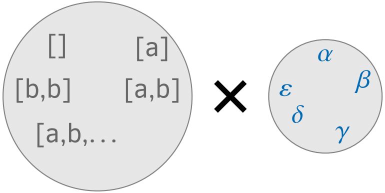


codatatype (*a*, *b*) *tllist* = TLNil *b* | TLCons *a* ((*a*, *b*) *tllist*)

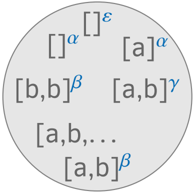


Terminated Lazy Lists

codatatype *a llist* $=$ LNil | LCons *a* (*a llist*)

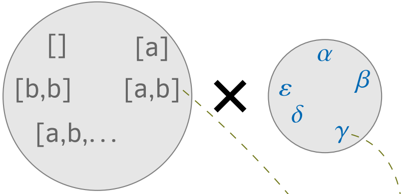


codatatype (*a*, *b*) *tllist* $=$ TLNil *b* | TLCons *a* ((*a*, *b*) *tllist*)

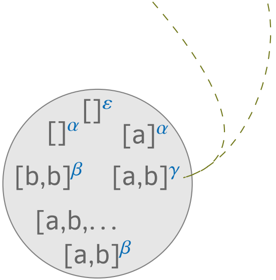


Terminated Lazy Lists

codatatype *a llist* = LNil | LCons *a* (*a llist*)

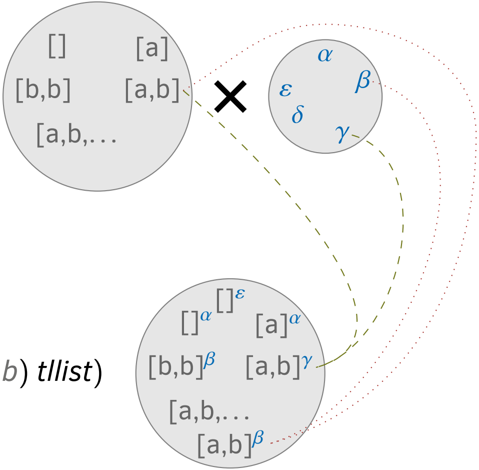


codatatype (*a*, *b*) *tllist* = TLNil *b* | TLCons *a* ((*a*, *b*) *tllist*)



Terminated Lazy Lists

codatatype *a llist* = LNil | LCons *a* (*a llist*)

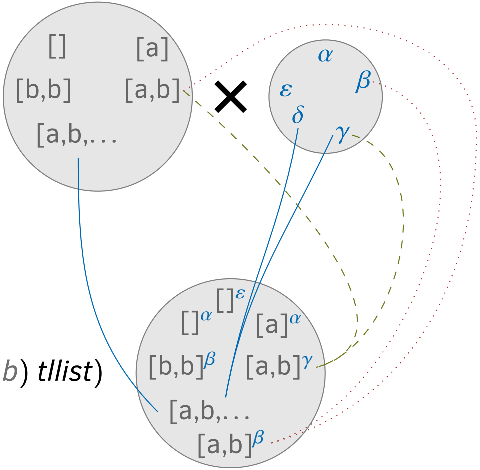


codatatype (*a*, *b*) *tllist* = TLNil *b* | TLCons *a* ((*a*, *b*) *tllist*)

Terminated Lazy Lists

codatatype *a llist* = LNil | LCons *a* (*a llist*)

codatatype (*a*, *b*) *tllist* = TLNil *b* | TLCons *a* ((*a*, *b*) *tllist*)

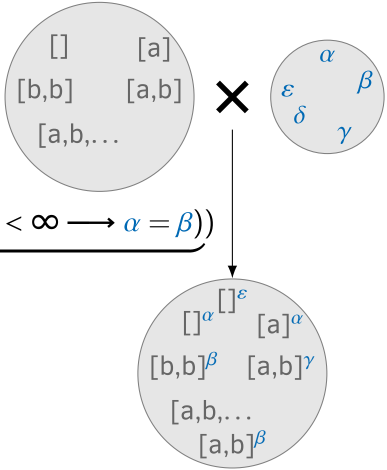


Terminated Lazy Lists

codatatype a *llist* $=$ LNil | LCons a (a *llist*)

quotient_type (a, b) *tllist* $=$

$$a\ \textit{llist} \times b\ /\ \underbrace{(\lambda(xs,\alpha)\ (ys,\beta).\ xs = ys \land (|xs| < \infty \longrightarrow \alpha = \beta))}_{\sim_{\text{tllist}}}$$

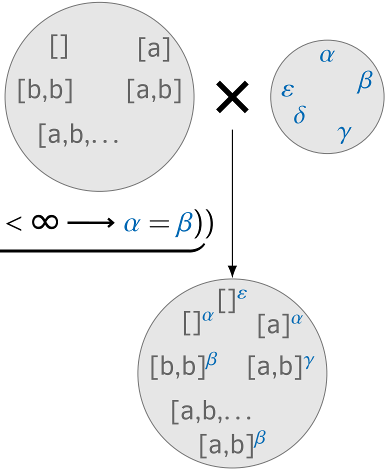


Terminated Lazy Lists

codatatype a *llist* = LNil | LCons a (a *llist*)

quotient_type (a, b) *tllist* =

$$a\ \mathit{llist} \times b\ /\ \underbrace{(\lambda(\mathsf{xs},\alpha)\ (\mathsf{ys},\beta).\ \mathsf{xs} = \mathsf{ys} \wedge (|\mathsf{xs}| < \infty \longrightarrow \alpha = \beta))}_{\sim_{\mathsf{tllist}}}$$



$(\mathsf{xs},\alpha) \sim_{\mathsf{tllist}} (\mathsf{ys},\beta) \longrightarrow \mathsf{set}_{\mathit{llist}}\ \mathsf{xs} = \mathsf{set}_{\mathit{llist}}\ \mathsf{ys}$

$(\mathsf{xs},\alpha) \sim_{\mathsf{tllist}} (\mathsf{ys},\beta) \longrightarrow \{\alpha\} = \{\beta\}$

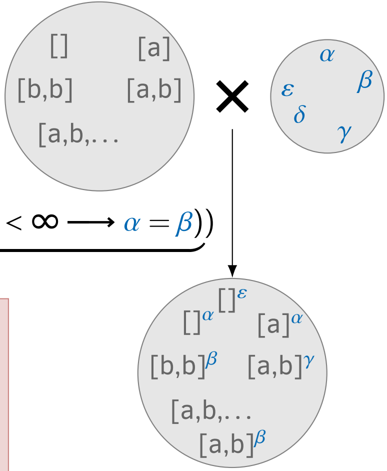
$\sim_{\mathsf{tllist}}$ preserves wide intersections

$\sim_{\mathsf{tllist}}$ preserves weak pullbacks

Terminated Lazy Lists

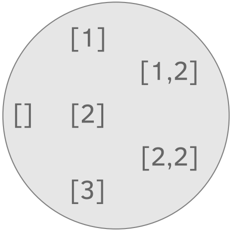
$$\textbf{codatatype}\ a\ \mathit{llist} = \mathrm{LNil} \mid \mathrm{LCons}\ a\ (a\ \mathit{llist})$$

$$\textbf{quotient\_type}\ (a, b)\ \mathit{tllist} =$$
$$a\ \mathit{llist} \times b\ /\ \underbrace{(\lambda(xs,\alpha)\ (ys,\beta).\ xs = ys \wedge (|xs| < \infty \longrightarrow \alpha = \beta))}_{\sim_{\text{tllist}}}$$



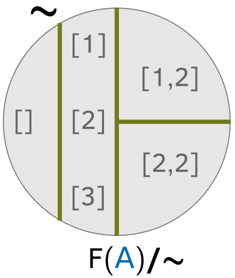
- ✓ $(xs,\alpha) \sim_{\text{tllist}} (ys,\beta) \longrightarrow \mathrm{set}_{\mathit{llist}}\ xs = \mathrm{set}_{\mathit{llist}}\ ys$
- ✗ $(xs,\alpha) \sim_{\text{tllist}} (ys,\beta) \longrightarrow \{\alpha\} = \{\beta\}$
- ✓ $\sim_{\text{tllist}}$ preserves wide intersections
- ✓ $\sim_{\text{tllist}}$ preserves weak pullbacks

How to correct?



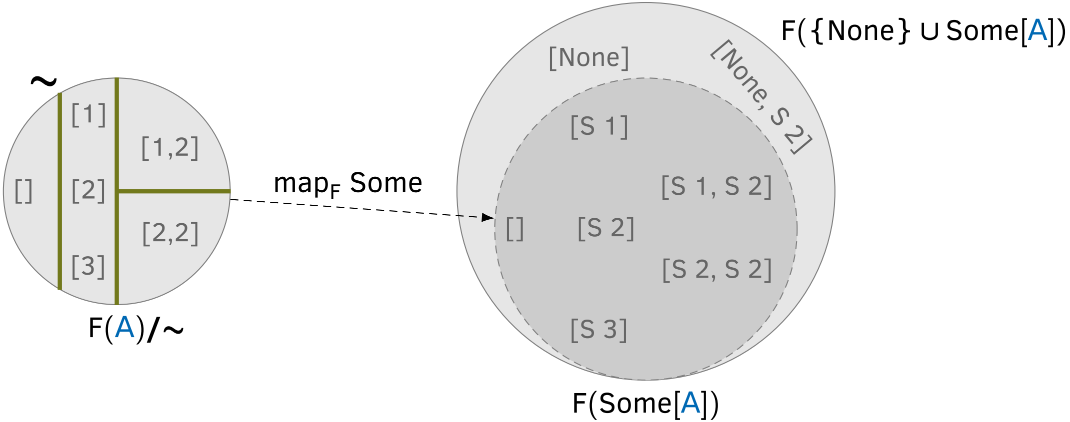
F(A)

How to correct?



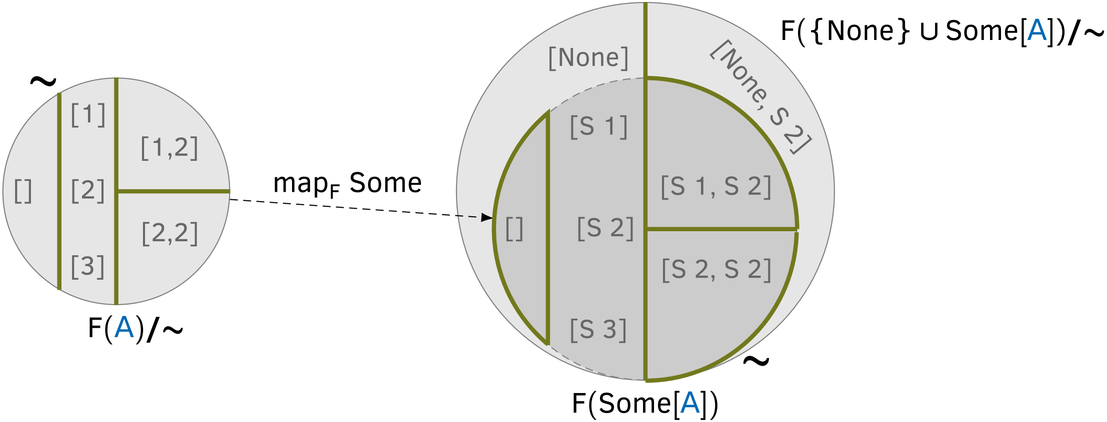
How to correct?

datatype *a option* = None | Some *a*



How to correct?

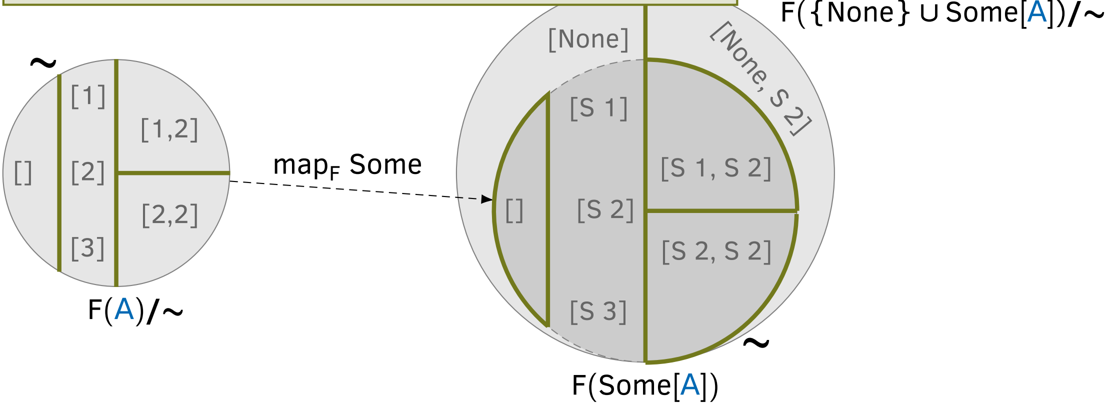
datatype *a option* = None | Some *a*



How to correct?

datatype a *option* = None | Some a

$$\mathrm{set}_{F/\sim}\ [x]_\sim = \bigcap_{y \in [\mathrm{map}_F\ \mathrm{Some}\ x]_\sim} \{a.\ \mathrm{Some}\ a \in \mathrm{set}_F\ y\}$$



Preservation theorem

- BNF F with equivalence relation $\sim$
- $\sim$ preserves wide intersections

$$\mathcal{A} \neq \{\} \land \bigcap \mathcal{A} \neq \{\} \longrightarrow \bigcap \{[A]_\sim | A \in F\langle\mathcal{A}\rangle\} \subseteq [\bigcap F\langle\mathcal{A}\rangle]_\sim$$

- $\sim$ weakly preserve pullbacks

$$R \bullet S \neq \{\} \longrightarrow \mathrm{rel}_F\ R \bullet \sim \bullet\ \mathrm{rel}_F\ S \subseteq \sim \bullet\ \mathrm{rel}_F\ (R \bullet S) \bullet \sim$$

Preservation theorem

- BNF F with equivalence relation $\sim$
- $\sim$ preserves wide intersections

$$\mathcal{A} \neq \{\} \wedge \bigcap \mathcal{A} \neq \{\} \longrightarrow \bigcap \{[A]_\sim \mid A \in F\langle\mathcal{A}\rangle\} \subseteq [\bigcap F\langle\mathcal{A}\rangle]_\sim$$

- $\sim$ weakly preserve pullbacks

$$R \bullet S \neq \{\} \longrightarrow \mathrm{rel}_F\ R \bullet \sim \bullet\ \mathrm{rel}_F\ S \subseteq \sim \bullet\ \mathrm{rel}_F\ (R \bullet S) \bullet \sim$$

yields BNF for F/$\sim$

- $\mathrm{map}_{F/\sim}\ f\ [x]_\sim = [\mathrm{map}_F\ f\ x]_\sim$
- $\mathrm{set}_{F/\sim}\ [x]_\sim = \bigcap_{y \in [\mathrm{map}_F\ \mathrm{Some}\ x]_\sim} \{a.\ \mathrm{Some}\ a \in \mathrm{set}_F\ y\}$
- $([x]_\sim, [y]_\sim) \in \mathrm{rel}_{F/\sim}\ R \longleftrightarrow (\mathrm{map}_F\ \mathrm{Some}\ x, \mathrm{map}_F\ \mathrm{Some}\ y) \in (\sim \bullet\ \mathrm{rel}_F\ (\mathrm{rel}_{option}\ R) \bullet \sim)$

lift_bnf in action

codatatype a *llist* $=$ LNil | LCons a (a *llist*)

definition $\sim_{\text{tllist}} :: a\ llist \times b \Rightarrow a\ llist \times b \Rightarrow bool$ **where**

$$(xs, \alpha) \sim_{\text{tllist}} (ys, \beta) \longleftrightarrow xs = ys \wedge (|xs| < \infty \longrightarrow \alpha = \beta)$$

quotient_type (a, b) *tllist* $= a\ llist \times b$ **/** $\sim_{\text{tllist}}$

lift_bnf (a, b) *tllist*

lift_bnf in action

codatatype a *llist* $=$ LNil | LCons a (a *llist*)

definition $\sim_{\text{tllist}} ::$ a *llist* $\times$ $b \Rightarrow a$ *llist* $\times$ $b \Rightarrow$ *bool* **where**

$$(xs, \alpha) \sim_{\text{tllist}} (ys, \beta) \longleftrightarrow xs = ys \wedge (|xs| < \infty \longrightarrow \alpha = \beta)$$

quotient_type (a, b) *tllist* $=$ a *llist* $\times$ b **/** $\sim_{\text{tllist}}$

lift_bnf (a, b) *tllist*

1. $A \bullet A' \neq \bot \longrightarrow B \bullet B' \neq \bot \longrightarrow$
   $\text{rel}_\times\ (\text{rel}_{\mathit{llist}}\ A)\ B \bullet \sim_{\text{tllist}} \bullet \text{rel}_\times\ (\text{rel}_{\mathit{llist}}\ A')\ B' \leq \sim_{\text{tllist}} \bullet \text{rel}_\times\ (\text{rel}_{\mathit{llist}}\ (A \bullet A'))\ (B \bullet B') \bullet \sim_{\text{tllist}}$
2. $S \neq \{\} \longrightarrow \bigcap S \neq \{\} \longrightarrow$
   $\bigcap_{A \in S} \{x.\ \exists y.\ y \sim_{\text{tllist}} x \wedge \text{set}_{\mathit{llist}}\ (\pi_1\ y) \subseteq A\} \subseteq \{x.\ \exists y.\ y \sim_{\text{tllist}} x \wedge \text{set}_{\mathit{llist}}\ (\pi_1\ y) \subseteq \bigcap S\}$
3. $S \neq \{\} \longrightarrow \bigcap S \neq \{\} \longrightarrow$
   $\bigcap_{A \in S} \{x.\ \exists y.\ y \sim_{\text{tllist}} x \wedge \pi_2\ y \in A\} \subseteq \{x.\ \exists y.\ y \sim_{\text{tllist}} x \wedge \pi_2\ y \in \bigcap S\}$

lift_bnf in action

codatatype a *llist* $=$ LNil | LCons a (a *llist*)

definition $\sim_{\text{tllist}}$ :: a *llist* × b ⇒ a *llist* × b ⇒ *bool* **where**

$(xs, \alpha) \sim_{\text{tllist}} (ys, \beta) \longleftrightarrow xs = ys \wedge (|xs| < \infty \longrightarrow \alpha = \beta)$

quotient_type (a, b) *tllist* $=$ a *llist* × b **/** $\sim_{\text{tllist}}$

lift_bnf (a, b) *tllist*

subgoal by (*auto* 0 4 *simp:* $\sim_{\text{tllist}}$_def ...)
subgoal by (*auto simp:* $\sim_{\text{tllist}}$_def)
subgoal by (*auto* 6 0 *simp:* $\sim_{\text{tllist}}$_def)
done

lift_bnf in action

codatatype a *llist* = LNil | LCons a (a *llist*)

definition $\sim_{\text{tllist}} ::$ a *llist* × b ⇒ a *llist* × b ⇒ *bool* **where**

$(\text{xs}, \alpha) \sim_{\text{tllist}} (\text{ys}, \beta) \longleftrightarrow \text{xs} = \text{ys} \wedge (|\text{xs}| < \infty \longrightarrow \alpha = \beta)$

quotient_type (a, b) *tllist* = a *llist* × b **/** $\sim_{\text{tllist}}$

lift_bnf (a, b) *tllist*

subgoal by (*auto* 0 4 *simp:* $\sim_{\text{tllist}}$ _def ...)
subgoal by (*auto simp:* $\sim_{\text{tllist}}$ _def)
subgoal by (*auto* 6 0 *simp:* $\sim_{\text{tllist}}$ _def)
done

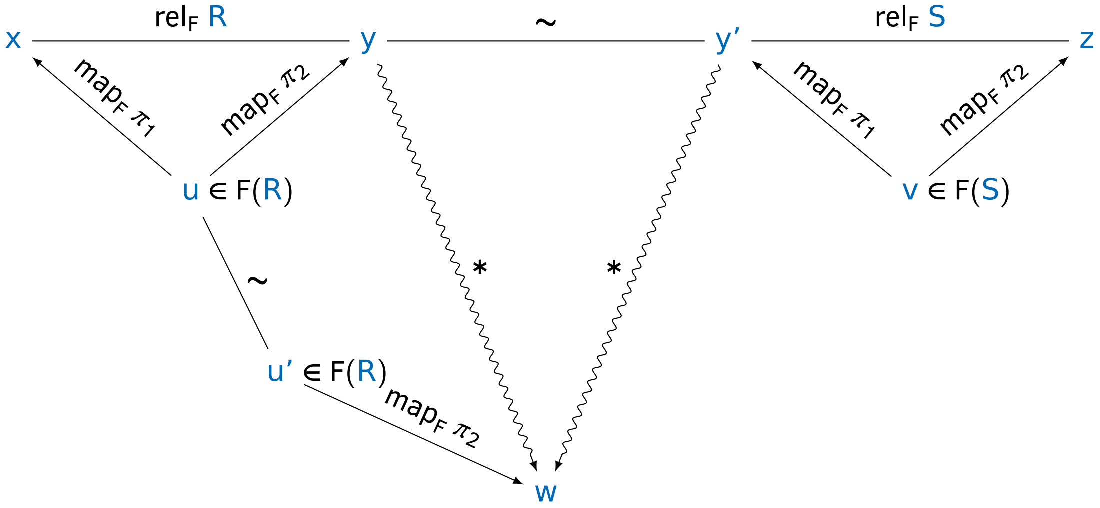
datatype *foo* = E | C ((*foo*, *foo*) *tllist*)

$$x \xrightarrow{\mathrm{rel}_F\ R} y \xrightarrow{\sim} y' \xrightarrow{\mathrm{rel}_F\ S} z$$

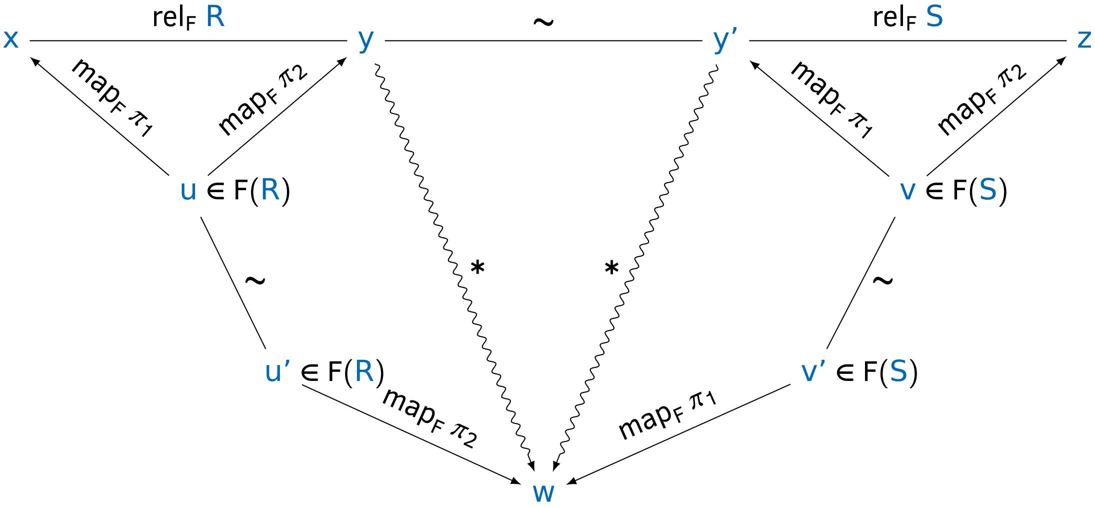
$$
\begin{tikzcd}
x \arrow[r, dash, "\mathrm{rel}_F\ R"] & y \arrow[r, dash, "\sim"] & y' \arrow[r, dash, "\mathrm{rel}_F\ S"] & z \\
u \in F(R) \arrow[u, "\mathrm{map}_F\ \pi_1"] \arrow[ur, "\mathrm{map}_F\ \pi_2"'] & & &
\end{tikzcd}
$$

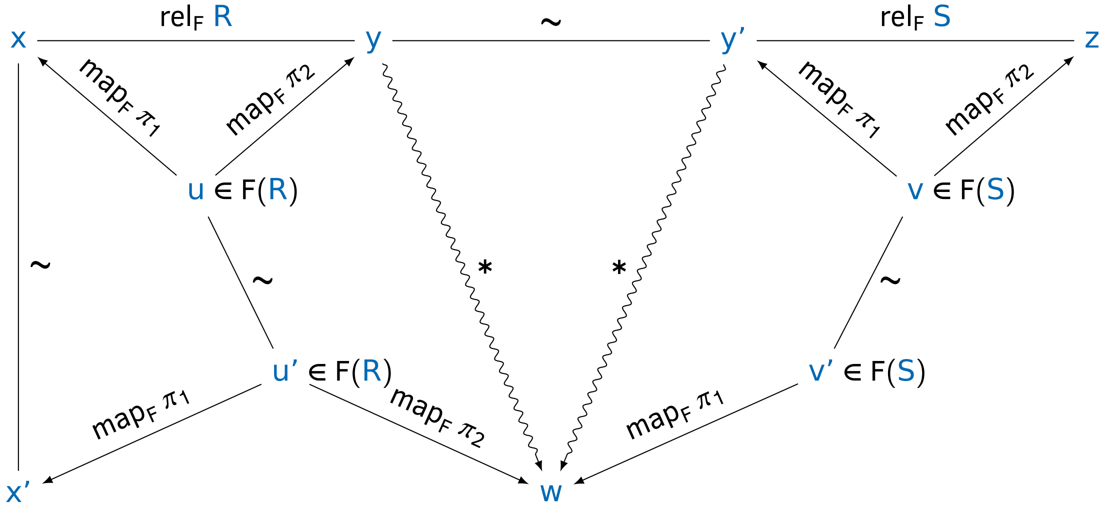
$$
\begin{tikzcd}
x \arrow[rr, dash, "\mathrm{rel}_F\ R"] & & y \arrow[r, dash, "\sim"] & y' \arrow[rr, dash, "\mathrm{rel}_F\ S"] & & z \\
& u \in F(R) \arrow[ul, "\mathrm{map}_F\ \pi_1"] \arrow[ur, "\mathrm{map}_F\ \pi_2"'] & & & v \in F(S) \arrow[ul, "\mathrm{map}_F\ \pi_1"] \arrow[ur, "\mathrm{map}_F\ \pi_2"'] &
\end{tikzcd}
$$

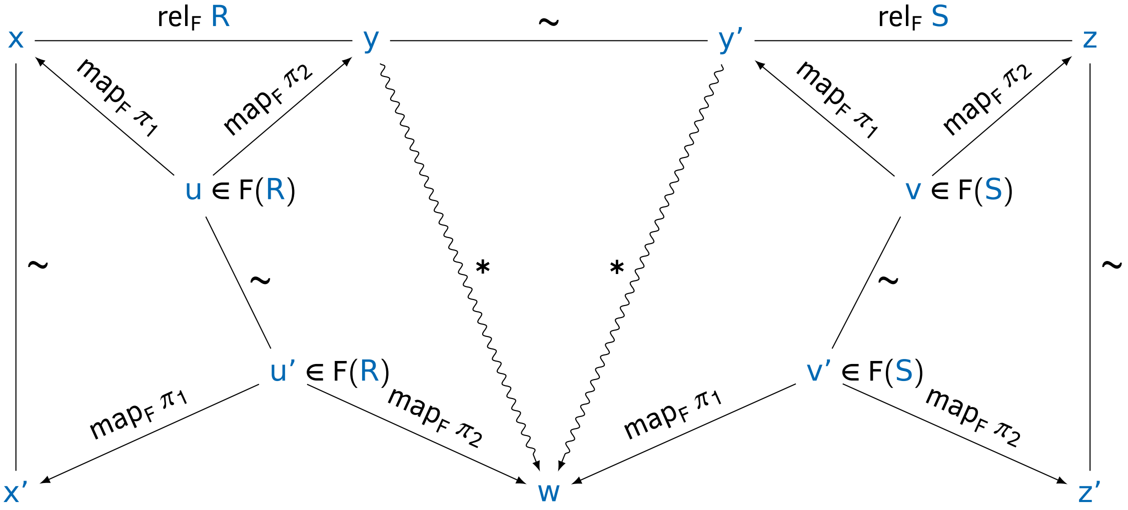
$$
\begin{tikzcd}
x \arrow[rr, dash, "\mathrm{rel}_F\ R"] & & y \arrow[rr, dash, "\sim"] \arrow[ddr, squiggly, "*"] & & y' \arrow[rr, dash, "\mathrm{rel}_F\ S"] \arrow[ddl, squiggly, "*"'] & & z \\
& u \in F(R) \arrow[ul, "\mathrm{map}_F\ \pi_1"] \arrow[ur, "\mathrm{map}_F\ \pi_2"'] & & & & v \in F(S) \arrow[ul, "\mathrm{map}_F\ \pi_1"] \arrow[ur, "\mathrm{map}_F\ \pi_2"'] & \\
& & & w & & &
\end{tikzcd}
$$

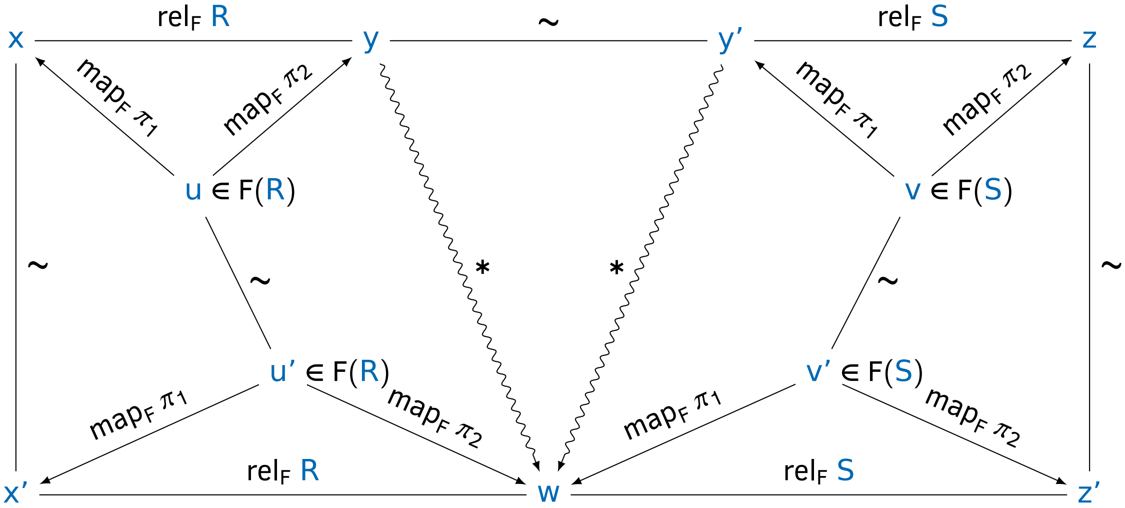


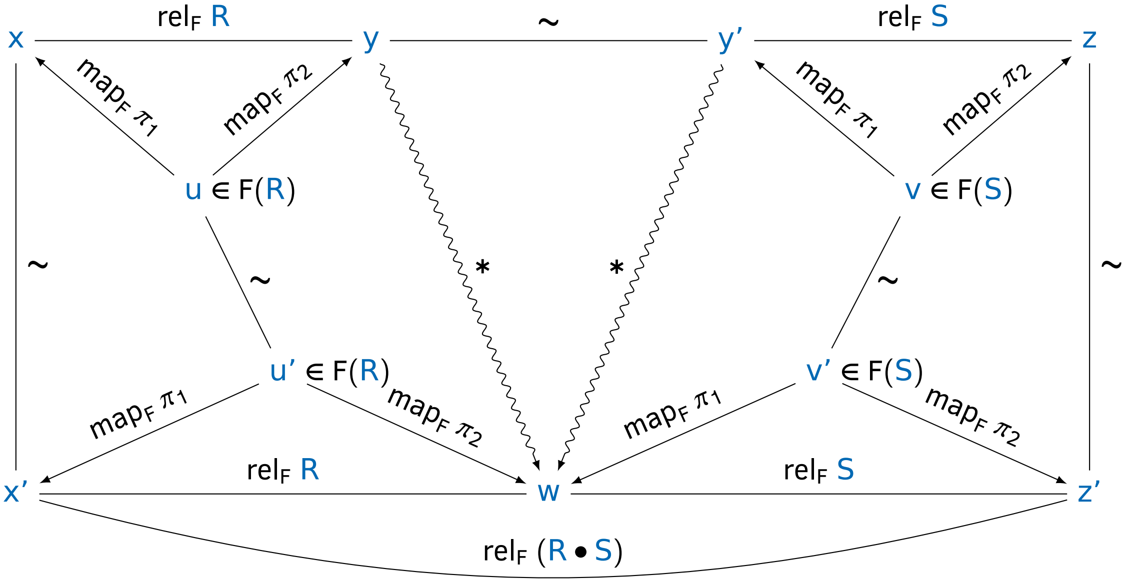
x
rel$_F$ R
y
~
y’
rel$_F$ S
z
map$_F$ π_1
map$_F$ π_2
map$_F$ π_1
map$_F$ π_2
u ∈ F(R)
v ∈ F(S)
~
*
*
u’ ∈ F(R)
map$_F$ π_2
w


x
rel_F R
y
~
y'
rel_F S
z
map_F π1
map_F π2
map_F π1
map_F π2
u ∈ F(R)
v ∈ F(S)
~
*
*
~
u' ∈ F(R)
v' ∈ F(S)
map_F π2
map_F π1
w


x
$\mathrm{rel}_F\ R$
y
~
y'
$\mathrm{rel}_F\ S$
z
$\mathrm{map}_F\ \pi_1$
$\mathrm{map}_F\ \pi_2$
$\mathrm{map}_F\ \pi_1$
$\mathrm{map}_F\ \pi_2$
$u \in F(R)$
$v \in F(S)$
~
~
*
*
~
$u' \in F(R)$
$v' \in F(S)$
$\mathrm{map}_F\ \pi_1$
$\mathrm{map}_F\ \pi_2$
$\mathrm{map}_F\ \pi_1$
x'
w


x
y
y'
z
$\mathrm{rel}_F\ R$
~
$\mathrm{rel}_F\ S$
$\mathrm{map}_F\ \pi_1$
$\mathrm{map}_F\ \pi_2$
$\mathrm{map}_F\ \pi_1$
$\mathrm{map}_F\ \pi_2$
$u \in F(R)$
$v \in F(S)$
~
~
~
~
*
*
$u' \in F(R)$
$v' \in F(S)$
$\mathrm{map}_F\ \pi_1$
$\mathrm{map}_F\ \pi_2$
$\mathrm{map}_F\ \pi_1$
$\mathrm{map}_F\ \pi_2$
x'
w
z'


x
rel_F R
y
~
y'
rel_F S
z
map_F π1
map_F π2
map_F π1
map_F π2
u ∈ F(R)
v ∈ F(S)
~
~
*
*
~
~
u' ∈ F(R)
v' ∈ F(S)
map_F π1
map_F π2
map_F π1
map_F π2
x'
rel_F R
w
rel_F S
z'



x
rel_F R
y
~
y'
rel_F S
z
map_F π1
map_F π2
map_F π1
map_F π2
u ∈ F(R)
v ∈ F(S)
~
~
*
*
~
~
u' ∈ F(R)
v' ∈ F(S)
map_F π1
map_F π2
map_F π1
map_F π2
x'
rel_F R
w
rel_F S
z'
rel_F (R • S)

Subdistributivity via rewrite relation

Sufficient conditions:

- BNF F with equivalence relation $\sim$
- $x \sim y \longrightarrow \mathrm{map}_F\ f\ x \sim \mathrm{map}_F\ f\ y \wedge \mathrm{set}_F\ x = \mathrm{set}_F\ y$

Subdistributivity via rewrite relation

Sufficient conditions:

- BNF F with equivalence relation $\sim$
- $x \sim y \longrightarrow \mathrm{map}_F\ f\ x \sim \mathrm{map}_F\ f\ y \wedge \mathrm{set}_F\ x = \mathrm{set}_F\ y$
- Rewrite relation $\rightsquigarrow$ over-approximates $\sim$
- $\rightsquigarrow$ confluent: $\begin{tikzcd} & x \arrow[dl, rightsquigarrow, "*"'] \arrow[dr, rightsquigarrow, "*"] & \\ y \arrow[dr, rightsquigarrow, "*"'] & & z \arrow[dl, rightsquigarrow, "*"] \\ & w & \end{tikzcd}$ and factors through projections: $\begin{tikzcd} x \arrow[r, "\mathrm{map}_F\ \pi_i"] \arrow[d, phantom, "\wr"] & x' \arrow[d, rightsquigarrow] \\ y \arrow[r] & y' \end{tikzcd}$

Subdistributivity via rewrite relation

Sufficient conditions:

- BNF F with equivalence relation $\sim$
- $x \sim y \longrightarrow \mathrm{map}_F\ f\ x \sim \mathrm{map}_F\ f\ y \wedge \mathrm{set}_F\ x = \mathrm{set}_F\ y$
- Rewrite relation $\rightsquigarrow$ over-approximates $\sim$
- $\rightsquigarrow$ confluent:
$$\begin{tikzcd} & x \arrow[dl, squiggly, "*"'] \arrow[dr, squiggly, "*"] & \\ y \arrow[dr, squiggly, "*"'] & & z \arrow[dl, squiggly, "*"] \\ & w & \end{tikzcd}$$
and factors through projections:
$$\begin{tikzcd} x \arrow[r, "\mathrm{map}_F\ \pi_i"] \arrow[d, phantom, "\wr"] & x' \arrow[d, squiggly] \\ y \arrow[r] & y' \end{tikzcd}$$

Distinct lists: $xs \cdot ys \rightsquigarrow xs \cdot [x] \cdot ys \quad \text{if} \quad x \in ys$

Proof effort: 50% shorter (58 instead of 126 lines)

a re_{ACI} is a BNF

inductive $\rightsquigarrow_{ACI}$ **where**

$$\frac{r \rightsquigarrow_{ACI} r' \quad s \rightsquigarrow_{ACI} s'}{\text{Alt } r\ s \rightsquigarrow_{ACI} \text{Alt } r'\ s'}$$

$$\frac{r \rightsquigarrow_{ACI} r' \quad s \rightsquigarrow_{ACI} s'}{\text{Conc } r\ s \rightsquigarrow_{ACI} \text{Conc } r'\ s'}$$

$$\frac{r \rightsquigarrow_{ACI} r'}{\text{Star } r \rightsquigarrow_{ACI} \text{Star } r'}$$

$r \rightsquigarrow_{ACI} r$

$r \rightsquigarrow_{ACI} \text{Alt } r\ r$

$\text{Alt } r\ s \rightsquigarrow_{ACI} \text{Alt } s\ r$

$\text{Alt } (\text{Alt } r\ s)\ t \rightsquigarrow_{ACI} \text{Alt } r\ (\text{Alt } s\ t)$

$\text{Alt } r\ (\text{Alt } s\ t) \rightsquigarrow_{ACI} \text{Alt } (\text{Alt } r\ s)\ t$

a re_{ACI} is a BNF

inductive $\rightsquigarrow_{\mathrm{ACI}}$ **where**

$$\frac{r \rightsquigarrow_{\mathrm{ACI}} r' \quad s \rightsquigarrow_{\mathrm{ACI}} s'}{\mathrm{Alt}\ r\ s \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ r'\ s'}$$

$$\frac{r \rightsquigarrow_{\mathrm{ACI}} r' \quad s \rightsquigarrow_{\mathrm{ACI}} s'}{\mathrm{Conc}\ r\ s \rightsquigarrow_{\mathrm{ACI}} \mathrm{Conc}\ r'\ s'}$$

$$\frac{r \rightsquigarrow_{\mathrm{ACI}} r'}{\mathrm{Star}\ r \rightsquigarrow_{\mathrm{ACI}} \mathrm{Star}\ r'}$$

$r \rightsquigarrow_{\mathrm{ACI}} r$

$r \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ r\ r$

$\mathrm{Alt}\ r\ s \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ s\ r$

$\mathrm{Alt}\ (\mathrm{Alt}\ r\ s)\ t \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ r\ (\mathrm{Alt}\ s\ t)$

$\mathrm{Alt}\ r\ (\mathrm{Alt}\ s\ t) \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ (\mathrm{Alt}\ r\ s)\ t$

- $(\rightsquigarrow_{\mathrm{ACI}} \cup \rightsquigarrow_{\mathrm{ACI}}^{-1})^* = (\sim_{\mathrm{ACI}})$
- $\rightsquigarrow_{\mathrm{ACI}}$ is confluent
- $\mathrm{map}_{re}\ \pi_1\ r \rightsquigarrow_{\mathrm{ACI}} s \longrightarrow \exists t.\ t \sim_{\mathrm{ACI}} r \wedge s = \mathrm{map}_{re}\ \pi_1\ t$
- $\mathrm{map}_{re}\ \pi_2\ r \rightsquigarrow_{\mathrm{ACI}} s \longrightarrow \exists t.\ t \sim_{\mathrm{ACI}} r \wedge s = \mathrm{map}_{re}\ \pi_2\ t$

$a\ re_{\mathrm{ACI}}$ is a BNF

inductive $\rightsquigarrow_{\mathrm{ACI}}$ **where**

$$\frac{r \rightsquigarrow_{\mathrm{ACI}} r' \quad s \rightsquigarrow_{\mathrm{ACI}} s'}{\mathrm{Alt}\ r\ s \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ r'\ s'}$$

$$\frac{r \rightsquigarrow_{\mathrm{ACI}} r' \quad s \rightsquigarrow_{\mathrm{ACI}} s'}{\mathrm{Conc}\ r\ s \rightsquigarrow_{\mathrm{ACI}} \mathrm{Conc}\ r'\ s'}$$

$$\frac{r \rightsquigarrow_{\mathrm{ACI}} r'}{\mathrm{Star}\ r \rightsquigarrow_{\mathrm{ACI}} \mathrm{Star}\ r'}$$

$r \rightsquigarrow_{\mathrm{ACI}} r$

$r \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ r\ r$

$\mathrm{Alt}\ r\ s \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ s\ r$

$\mathrm{Alt}\ (\mathrm{Alt}\ r\ s)\ t \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ r\ (\mathrm{Alt}\ s\ t)$

$\mathrm{Alt}\ r\ (\mathrm{Alt}\ s\ t) \rightsquigarrow_{\mathrm{ACI}} \mathrm{Alt}\ (\mathrm{Alt}\ r\ s)\ t$

- $(\rightsquigarrow_{\mathrm{ACI}} \cup \rightsquigarrow_{\mathrm{ACI}}^{-1})^* = (\sim_{\mathrm{ACI}})$
- $\rightsquigarrow_{\mathrm{ACI}}$ is confluent
- $\mathrm{map}_{re}\ \pi_1\ r \rightsquigarrow_{\mathrm{ACI}} s \longrightarrow \exists t.\ t \sim_{\mathrm{ACI}} r \wedge s = \mathrm{map}_{re}\ \pi_1\ t$
- $\mathrm{map}_{re}\ \pi_2\ r \rightsquigarrow_{\mathrm{ACI}} s \longrightarrow \exists t.\ t \sim_{\mathrm{ACI}} r \wedge s = \mathrm{map}_{re}\ \pi_2\ t$

lift_bnf $a\ re_{\mathrm{ACI}}\ \langle proof \rangle$

$\alpha\ re_{ACI}$ is a BNF

inductive $\rightsquigarrow_{ACI}$ **where**

$$\frac{r \rightsquigarrow_{ACI} r' \quad s \rightsquigarrow_{ACI} s'}{\mathrm{Alt}\ r\ s \rightsquigarrow_{ACI} \mathrm{Alt}\ r'\ s'}$$

$$\frac{r \rightsquigarrow_{ACI} r' \quad s \rightsquigarrow_{ACI} s'}{\mathrm{Conc}\ r\ s \rightsquigarrow_{ACI} \mathrm{Conc}\ r'\ s'}$$

$$\frac{r \rightsquigarrow_{ACI} r'}{\mathrm{Star}\ r \rightsquigarrow_{ACI} \mathrm{Star}\ r'}$$

$r \rightsquigarrow_{ACI} r$

$r \rightsquigarrow_{ACI} \mathrm{Alt}\ r\ r$

$\mathrm{Alt}\ r\ s \rightsquigarrow_{ACI} \mathrm{Alt}\ s\ r$

$\mathrm{Alt}\ (\mathrm{Alt}\ r\ s)\ t \rightsquigarrow_{ACI} \mathrm{Alt}\ r\ (\mathrm{Alt}\ s\ t)$

$\mathrm{Alt}\ r\ (\mathrm{Alt}\ s\ t) \rightsquigarrow_{ACI} \mathrm{Alt}\ (\mathrm{Alt}\ r\ s)\ t$

- $(\rightsquigarrow_{ACI} \cup \rightsquigarrow_{ACI}^{-1})^* = (\sim_{ACI})$
- $\rightsquigarrow_{ACI}$ is confluent
- $\mathrm{map}_{re}\ \pi_1\ r \rightsquigarrow_{ACI} s \longrightarrow \exists t.\ t \sim_{ACI} r \wedge s = \mathrm{map}_{re}\ \pi_1\ t$
- $\mathrm{map}_{re}\ \pi_2\ r \rightsquigarrow_{ACI} s \longrightarrow \exists t.\ t \sim_{ACI} r \wedge s = \mathrm{map}_{re}\ \pi_2\ t$

lift_bnf *α re*$_{ACI}$ ⟨*proof*⟩

datatype *ldl* = Prop *string* | And *ldl ldl* | Neg *ldl*
| Match (*ldl re*$_{ACI}$)

Epilogue

lift_bnf

- part of Isabelle2020
- 1600 lines of Isabelle/ML
- generation of transfer rules

Epilogue

lift_bnf

- part of Isabelle2020
- 1600 lines of Isabelle/ML
- generation of transfer rules

Applications

- (co)datatypes
- Lifting and Transfer
- QPF

Epilogue

lift_bnf

- part of Isabelle2020
- 1600 lines of Isabelle/ML
- generation of transfer rules

Applications

- (co)datatypes
- Lifting and Transfer
- QPF

Limitations

- terms modulo α-equivalence
  [Blanchette, Gheri, Popescu, T., POPL'19]
- signed multisets
  [Blanchette, Fleury, T., FSCD'16]

Epilogue

lift_bnf

- part of Isabelle2020
- 1600 lines of Isabelle/ML
- generation of transfer rules

Applications

- (co)datatypes
- Lifting and Transfer
- QPF

Limitations

- terms modulo α-equivalence
  [Blanchette, Gheri, Popescu, T., POPL'19]
- signed multisets
  [Blanchette, Fleury, T., FSCD'16]

Future Work

- partial quotients
- generalizations of BNFs
  [L., S., ITP'18]
  [Blanchette, Gheri, Popescu, T., POPL'19]

Quotients of Bounded Natural Functors

Basil Fürer

Andreas Lochbihler

Digital Asset

ETH zürich

Joshua Schneider

Dmitriy Traytel

merci!
questions?