Extracting nested relational queries from implicit definitions

Pierre Pradic
(j.w.w. Michael Benedikt)

University of Oxford

December 4^{th}, 2020

Plan of the talk

- The nested relational calculus (NRC)
- Implicit definability, implicit $\rightarrow$ explicit for the flat case
- Our contribution: implicit $\rightarrow$ explicit for NRC

Outline

The nested relational calculus (NRC)

Syntax

Types: $T, U ::= \mathfrak{U} \mid \mathsf{Set}(T) \mid 1 \mid T \times U$

Terms: $Q, R ::= x \mid \emptyset \mid Q \cup R \mid Q \setminus R \mid \{Q\} \mid \bigcup\{Q \mid x \in R\} \mid \langle Q, \ldots, R \rangle \mid \pi_i$

every variable x carries a type T

Terms represent *nested queries* of some given type $T \to U$

- Cartesian structure $\pi_i, \langle \ldots \rangle$
- Monad structure on Set $\{-\}, \bigcup$
- Idempotent monoid $\mathsf{Set}(T)$ $\emptyset, \cup$
- Set difference $Q \setminus R$

Generalizes flat relational queries with higher-order types

$$\text{flat} \quad \cong \quad \mathsf{Set}(\mathfrak{U}^{i_1}) \times \ldots \times \mathsf{Set}(\mathfrak{U}^{i_k}) \to \mathsf{Set}(\mathfrak{U}^m)$$

Examples

A flat query

The fiber of a relation f at some point x

$$\begin{array}{rcl} \mathrm{fib}: \quad \mathfrak{U} \times \mathrm{Set}(\mathfrak{U} \times \mathfrak{U}) & \to & \mathrm{Set}(\mathfrak{U}) \\ (x, f) & \mapsto & f^{-1}(x) \end{array}$$

- "concrete instance": $\mathfrak{U}$ contains names, f="is the parent of"
- can be written as $(x, f) \mapsto \bigcup\{\mathrm{case}(\pi_2(p) =_{\mathfrak{U}} x, \{\pi_1(p)\}, \emptyset) \mid p \in f\}$

syntactic sugar: case, $=_{\mathfrak{U}}$

Examples

A flat query

The fiber of a relation f at some point x

$$\begin{array}{rccc} \mathsf{fib}: & \mathfrak{U} \times \mathsf{Set}(\mathfrak{U} \times \mathfrak{U}) & \to & \mathsf{Set}(\mathfrak{U}) \\ & (x, f) & \mapsto & f^{-1}(x) \end{array}$$

- "concrete instance": $\mathfrak{U}$ contains names, f="is the parent of"
- can be written as $(x, f) \mapsto \bigcup\{\mathsf{case}(\pi_2(p) =_{\mathfrak{U}} x, \{\pi_1(p)\}, \emptyset) \mid p \in f\}$

syntactic sugar: case, $=_{\mathfrak{U}}$

A genuine nested query

Collect all fibers of f

$$\begin{array}{rccc} \mathsf{fibs}: & \mathsf{Set}(\mathfrak{U} \times \mathfrak{U}) & \to & \mathsf{Set}(\mathfrak{U} \times \mathsf{Set}(\mathfrak{U})) \\ & f & \mapsto & \{(a, f^{-1}(a)) \mid a \in \mathsf{cod}(f)\} \end{array}$$

- can be written as $f \mapsto \bigcup\{\{\mathsf{fib}(x, f)\} \mid x \in \{\pi_1(p) \mid p \in f\}\}$

Expressiveness of NRC

From now on, set Bool $:=$ Set(1).
Derivable constructs:

- maps $\{Q(x) \mid x \in R\}$
- set intersection $Q \cap R$
- case analyses if the output is some Set(T)
- basic predicates $=_T \colon T \times T \to \text{Bool}$, $\in_T \colon T \times \text{Set}(T) \to \text{Bool}$

Expressiveness of NRC

From now on, set $\mathsf{Bool} := \mathsf{Set}(1)$.
Derivable constructs:

- maps $\{Q(x) \mid x \in R\}$
- set intersection $Q \cap R$
- case analyses — if the output is some $\mathsf{Set}(T)$
- basic predicates $=_T \colon T \times T \to \mathsf{Bool}$, $\in_T \colon T \times \mathsf{Set}(T) \to \mathsf{Bool}$

Proposition

NRC queries $Q(x^T) : T \to \mathsf{Bool}$ correspond exactly to Δ_0 formulas $\varphi(x^T)$.

Expressiveness of NRC

From now on, set Bool := Set(1).
Derivable constructs:

- maps $\{Q(x) \mid x \in R\}$
- set intersection $Q \cap R$
- case analyses — if the output is some Set(T)
- basic predicates $=_T$: $T \times T \to$ Bool, $\in_T$: $T \times$ Set(T) $\to$ Bool

Proposition

NRC queries $Q(x^T) : T \to$ Bool correspond exactly to Δ_0 formulas $\varphi(x^T)$.

$\rightsquigarrow$ Δ_0-separation is encodable in NRC

$$\{x \in Q \mid \varphi(x)\}$$

Limits to the expressiveness of NRC

For practical purposes, NRC is not be too expressive

- NRC is *conservative* over idealized SQL — i.e., for flat queries
- for finite inputs, the output has *polynomial* size

Consequences

- rules out $x \mapsto \mathcal{P}(x)$
- rules out *curryfication!* — Consider $(x, y) \mapsto \texttt{tt}$

$$[T \to \mathsf{Set}(U)] \not\cong [T \times U \to \mathsf{Bool}]$$

$$[T \to \mathsf{Set}(U)] \hookrightarrow [T \times U \to \mathsf{Bool}]$$

(For the rest of the talk: no finiteness assumptions)

Outline

Implicit definitions

Implicit definability

$\varphi(i, o)$ is a *functional* definition of o in terms of i if

$$\varphi(i, o) \wedge \varphi(i, o') \quad \Rightarrow \quad o = o'$$

Defines a *partial* function $I \to O$

Implicit definability

$\varphi(i, o)$ is a *functional* definition of o in terms of i if

$$\varphi(i, o) \wedge \varphi(i, o') \quad \Rightarrow \quad o = o'$$

Defines a *partial* function $I \to O$

Main theorem

Expressible in NRC $\quad \Longleftrightarrow \quad$ Has an implicit definition

- We call a NRC term an *explicit definition*
- Partial implicit definitions $\to$ compatible total explicit definitions
- (Orthogonal to C-H approaches, where *totality* proofs are used)

$\Rightarrow$: easy to map a NRC expression to an implicit definition

Use-case for implicit$\rightarrow$explicit

Main theorem

Expressible in NRC $\iff$ Has an implicit definition

Implicit definitions might arguably be more convenient for users at times.

Main theorem

Expressible in NRC $\iff$ Has an implicit definition

Implicit definitions might arguably be more convenient for users at times.

Use-case: inverting a query

Consider an *injective* NRC query such as fibs

$$\begin{array}{rccl} \mathrm{fibs}: & \mathrm{Set}(\mathfrak{U} \times \mathfrak{U}) & \to & \mathrm{Set}(\mathfrak{U} \times \mathrm{Set}(\mathfrak{U})) \\ & f & \mapsto & \{(a, f^{-1}(a)) \mid a \in \mathrm{cod}(f)\} \end{array}$$

Use-case for implicit$\to$explicit

Main theorem

Expressible in NRC $\iff$ Has an implicit definition

Implicit definitions might arguably be more convenient for users at times.

Use-case: inverting a query

Consider an *injective* NRC query such as fibs

$$\begin{array}{rccc} \mathsf{fibs}: & \mathsf{Set}(\mathfrak{U} \times \mathfrak{U}) & \to & \mathsf{Set}(\mathfrak{U} \times \mathsf{Set}(\mathfrak{U})) \\ & f & \mapsto & \{(a, f^{-1}(a)) \mid a \in \mathsf{cod}(f)\} \end{array}$$

- can be converted to an implicit $\varphi(f, F)$

Use-case for implicit→explicit

Main theorem

Expressible in NRC $\iff$ Has an implicit definition

Implicit definitions might arguably be more convenient for users at times.

Use-case: inverting a query

Consider an *injective* NRC query such as fibs

$$\begin{array}{rccl} \mathsf{fibs}: & \mathsf{Set}(\mathfrak{U} \times \mathfrak{U}) & \to & \mathsf{Set}(\mathfrak{U} \times \mathsf{Set}(\mathfrak{U})) \\ & f & \mapsto & \{(a, f^{-1}(a)) \mid a \in \mathsf{cod}(f)\} \end{array}$$

- can be converted to an implicit $\varphi(f, F)$
- $\varphi(f, F)$ defines a partial function $F \mapsto f$

Use-case for implicit→explicit

Main theorem

Expressible in NRC $\iff$ Has an implicit definition

Implicit definitions might arguably be more convenient for users at times.

Use-case: inverting a query

Consider an *injective* NRC query such as fibs

$$\begin{array}{rccc} \mathrm{fibs}: & \mathrm{Set}(\mathfrak{U} \times \mathfrak{U}) & \to & \mathrm{Set}(\mathfrak{U} \times \mathrm{Set}(\mathfrak{U})) \\ & f & \mapsto & \{(a, f^{-1}(a)) \mid a \in \mathrm{cod}(f)\} \end{array}$$

- can be converted to an implicit $\varphi(f, F)$
- $\varphi(f, F)$ defines a partial function $F \mapsto f$

$\rightsquigarrow$ a NRC-definable retract of fibs

Interpolation

The result was already known for the flat case.

Beth definability

Let $\varphi(R)$ be a first-order formula.
If $\varphi(R) \wedge \varphi(R') \Rightarrow R \equiv R'$, then there is a FO $\psi(\vec{x})$ such that $\varphi(\psi)$.

i.e., R first-order definable

Interpolation

The result was already known for the flat case.

Beth definability

Let $\varphi(R)$ be a first-order formula.
If $\varphi(R) \wedge \varphi(R') \Rightarrow R \equiv R'$, then there is a FO $\psi(\vec{x})$ such that $\varphi(\psi)$.

i.e., R first-order definable

- Model-theoretic proof using amalgamation

Interpolation

The result was already known for the flat case.

Beth definability

Let $\varphi(R)$ be a first-order formula.
If $\varphi(R) \wedge \varphi(R') \Rightarrow R \equiv R'$, then there is a FO $\psi(\vec{x})$ such that $\varphi(\psi)$.

i.e., R first-order definable

- Model-theoretic proof using amalgamation
- Proof-theoretic effective proof using interpolation

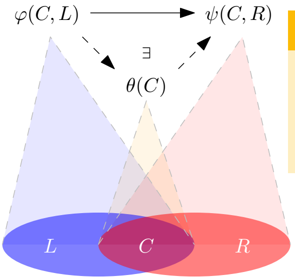
The result was already known for the flat case.

Beth definability

Let $\varphi(R)$ be a first-order formula.
If $\varphi(R) \wedge \varphi(R') \Rightarrow R \equiv R'$, then there is a FO $\psi(\vec{x})$ such that $\varphi(\psi)$.

i.e., R first-order definable

- Model-theoretic proof using amalgamation
- Proof-theoretic effective proof using interpolation



Craig interpolation

If $\varphi \Rightarrow \psi$, there exists θ such that

$$\varphi \Rightarrow \theta \qquad \text{and} \qquad \theta \Rightarrow \psi$$

and Vocabulary$(\theta) \subseteq$ Vocabulary$(\varphi) \cap$ Vocabulary(ψ)

Interpolation

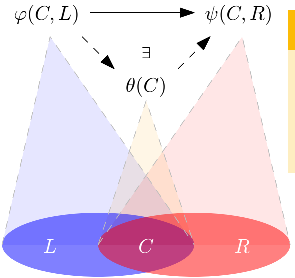
The result was already known for the flat case.

Beth definability

Let $\varphi(R)$ be a first-order formula.
If $\varphi(R) \wedge \varphi(R') \Rightarrow R \equiv R'$, then there is a FO $\psi(\vec{x})$ such that $\varphi(\psi)$.

i.e., R first-order definable

- Model-theoretic proof using amalgamation
- Proof-theoretic effective proof using interpolation



Craig interpolation

If $\varphi \Rightarrow \psi$, there exists θ such that

$$\varphi \Rightarrow \theta \qquad \text{and} \qquad \theta \Rightarrow \psi$$

and Vocabulary$(\theta) \subseteq$ Vocabulary$(\varphi) \cap$ Vocabulary(ψ)

- θ linear-time computable from a cut-free derivation
- Rather robust result

Δ_0-interpolation, intuitionistic/linear logic...

Proof idea for the flat case

Fix an implicit definition $\varphi(I, O)$ with $I : \mathsf{Set}(\mathfrak{U}^k)$ and $O : \mathsf{Set}(\mathfrak{U}^m)$.

Effective proof sketch

Proof idea for the flat case

Fix an implicit definition $\varphi(I, O)$ with $I : \mathsf{Set}(\mathfrak{U}^k)$ and $O : \mathsf{Set}(\mathfrak{U}^m)$.

Effective proof sketch

1. Apply interpolation to
$$\varphi(I, O) \wedge O(\vec{x}) \quad \vdash \quad \varphi(I, O') \Rightarrow O'(\vec{x})$$
to obtain an explicit Δ_0 definition $\theta(I, \vec{x})$.

Proof idea for the flat case

Fix an implicit definition $\varphi(I, O)$ with $I : \mathsf{Set}(\mathfrak{U}^k)$ and $O : \mathsf{Set}(\mathfrak{U}^m)$.

Effective proof sketch

1. Apply interpolation to
$$\varphi(I, O) \wedge O(\vec{x}) \quad \vdash \quad \varphi(I, O') \Rightarrow O'(\vec{x})$$
to obtain an explicit Δ_0 definition $\theta(I, \vec{x})$.
2. There is a NRC term $M : \mathsf{Set}(\mathfrak{U}^k) \to \mathsf{Set}(\mathfrak{U}^m)$ *maximal for* $\subseteq$

Proof idea for the flat case

Fix an implicit definition $\varphi(I, O)$ with $I : \mathsf{Set}(\mathfrak{U}^k)$ and $O : \mathsf{Set}(\mathfrak{U}^m)$.

Effective proof sketch

1. Apply interpolation to
$$\varphi(I, O) \wedge O(\vec{x}) \quad \vdash \quad \varphi(I, O') \Rightarrow O'(\vec{x})$$
to obtain an explicit Δ_0 definition $\theta(I, \vec{x})$.
2. There is a NRC term $M : \mathsf{Set}(\mathfrak{U}^k) \to \mathsf{Set}(\mathfrak{U}^m)$ *maximal for* $\subseteq$
Additionally, $\theta(I, \vec{x}) \Leftrightarrow \theta^M(I, \vec{x})$ for any $\theta \in \Delta_0$.

Proof idea for the flat case

Fix an implicit definition $\varphi(I, O)$ with $I : \mathsf{Set}(\mathfrak{U}^k)$ and $O : \mathsf{Set}(\mathfrak{U}^m)$.

Effective proof sketch

1. Apply interpolation to
$$\varphi(I, O) \wedge O(\vec{x}) \quad \vdash \quad \varphi(I, O') \Rightarrow O'(\vec{x})$$
to obtain an explicit Δ_0 definition $\theta(I, \vec{x})$.
2. There is a NRC term $M : \mathsf{Set}(\mathfrak{U}^k) \to \mathsf{Set}(\mathfrak{U}^m)$ *maximal for* $\subseteq$
Additionally, $\theta(I, \vec{x}) \Leftrightarrow \theta^M(I, \vec{x})$ for any $\theta \in \Delta_0$.
3. Conclude using Δ_0-comprehension in NRC
$$\{\vec{x} \in M \mid \theta(I, x)\}$$

Proof idea for the flat case

Fix an implicit definition $\varphi(I, O)$ with $I : \mathsf{Set}(\mathfrak{U}^k)$ and $O : \mathsf{Set}(\mathfrak{U}^m)$.

Effective proof sketch

1. Apply interpolation to
$$\varphi(I, O) \wedge O(\vec{x}) \quad \vdash \quad \varphi(I, O') \Rightarrow O'(\vec{x})$$
to obtain an explicit Δ_0 definition $\theta(I, \vec{x})$.
2. There is a NRC term $M : \mathsf{Set}(\mathfrak{U}^k) \to \mathsf{Set}(\mathfrak{U}^m)$ *maximal for* $\subseteq$
Additionally, $\theta(I, \vec{x}) \Leftrightarrow \theta^M(I, \vec{x})$ for any $\theta \in \Delta_0$.
3. Conclude using Δ_0-comprehension in NRC
$$\{\vec{x} \in M \mid \theta(I, x)\}$$

Difficulty with the nested case: there is no M!

Outline

Main ineffective result

Main theorem

Expressible in NRC $\iff$ Has a Δ_0 implicit definition

$\rightsquigarrow$ automatic translation of implicit definitions to NRC?

Main ineffective result

Main theorem

Expressible in NRC $\iff$ Has a Δ_0 implicit definition

$\rightsquigarrow$ automatic translation of implicit definitions to NRC?

Problem: a non-constructive proof

- Model-theoretic argument — omitting types, ...
- a generalization of Beth for multi-sorted structures

Main ineffective result

Main theorem

Expressible in NRC $\iff$ Has a Δ_0 implicit definition

$\rightsquigarrow$ automatic translation of implicit definitions to NRC?

Problem: a non-constructive proof

- Model-theoretic argument — omitting types, ...
- a generalization of Beth for multi-sorted structures

Partial effective result

Expressible in NRC $\iff$ Has an **intuitionistic** Δ_0 implicit definition

Main effective result

Partial effective result

Expressible in NRC $\iff$ Has an **intuitionistic** Δ_0 implicit definition

Main effective result

Partial effective result

Expressible in NRC $\iff$ Has an **intuitionistic** Δ_0 implicit definition

Algorithmic content

Input:

- An implicit definition $\varphi(i, o)$
- An **intuitionistic** (cut-free) proof π of functionality of φ

Output:

- A NRC query $Q(i)$ such that $\varphi(i, o) \Rightarrow Q(i) = o$

- Linear-time

Caveat: cut-elimination

Main effective result

Partial effective result

Expressible in NRC $\iff$ Has an **intuitionistic** Δ_0 implicit definition

Algorithmic content

Input:

- An implicit definition $\varphi(i, o)$
- An **intuitionistic** (cut-free) proof π of functionality of φ

Output:

- A NRC query $Q(i)$ such that $\varphi(i, o) \Rightarrow Q(i) = o$

- Linear-time

Caveat: cut-elimination

- Let's look at the details...

Δ_0 formulas and intuitionistic sequents

Let's make several quality-of-life adjustments

$$\begin{array}{lcl} t, u & ::= & x \mid (t, u) \mid \pi_1(t) \mid \pi_2(t) \mid () \\ \varphi, \psi & ::= & t =_{\mathfrak{U}} u \mid t \neq_{\mathfrak{U}} u \mid \exists x \in t\ \varphi \mid \forall x \in t\ \varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \end{array}$$

Δ_0 formulas and intuitionistic sequents

Let's make several quality-of-life adjustments

$$
\begin{array}{lcl}
t, u & ::= & x \mid (t, u) \mid \pi_1(t) \mid \pi_2(t) \mid () \\
\varphi, \psi & ::= & t =_{\mathfrak{U}} u \mid t \neq_{\mathfrak{U}} u \mid \exists x \in t\ \varphi \mid \forall x \in t\ \varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi
\end{array}
$$

We use a cut-free version of LJ as our proof system. Cut is admissible

Δ_0 formulas and intuitionistic sequents

Let's make several quality-of-life adjustments

$$
\begin{array}{lcl}
t, u & ::= & x \mid (t, u) \mid \pi_1(t) \mid \pi_2(t) \mid () \\
\varphi, \psi & ::= & t =_{\mathfrak{U}} u \mid t \neq_{\mathfrak{U}} u \mid \exists x \in t\ \varphi \mid \forall x \in t\ \varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi
\end{array}
$$

We use a cut-free version of LJ as our proof system. Cut is admissible

Derived formulas

$$
\begin{array}{rcl}
t =_{\mathsf{Set}(T)} u & := & t \subseteq_T u \quad \wedge \quad u \subseteq_T t \\
t \subseteq_T u & := & \forall x \in t.\ x \in_T u \\
t \in_T u & := & \exists x \in u.\ t =_T u
\end{array}
$$

Δ_0 formulas and intuitionistic sequents

Let's make several quality-of-life adjustments

$$
\begin{array}{lcl}
t, u & ::= & x \mid (t, u) \mid \pi_1(t) \mid \pi_2(t) \mid () \\
\varphi, \psi & ::= & t =_{\mathfrak{U}} u \mid t \neq_{\mathfrak{U}} u \mid \exists x \in t\ \varphi \mid \forall x \in t\ \varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi
\end{array}
$$

We use a cut-free version of LJ as our proof system. Cut is admissible

Derived formulas

$$
\begin{array}{rcl}
t =_{\mathsf{Set}(T)} u & := & t \subseteq_T u \quad \wedge \quad u \subseteq_T t \\
t \subseteq_T u & := & \forall x \in t.\ x \in_T u \\
t \in_T u & := & \exists x \in u.\ t =_T u
\end{array}
$$

- Allows to suppress the axiom of extensionality
- **No further set-theoretic axioms!**

Δ_0 formulas and intuitionistic sequents

Let's make several quality-of-life adjustments

$$
\begin{array}{lcl}
t, u & ::= & x \mid (t, u) \mid \pi_1(t) \mid \pi_2(t) \mid () \\
\varphi, \psi & ::= & t =_{\mathfrak{U}} u \mid t \neq_{\mathfrak{U}} u \mid \exists x \in t\ \varphi \mid \forall x \in t\ \varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi
\end{array}
$$

We use a cut-free version of LJ as our proof system. Cut is admissible

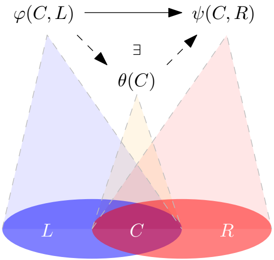
Derived formulas

$$
\begin{array}{rcl}
t =_{\mathsf{Set}(T)} u & := & t \subseteq_T u \quad \wedge \quad u \subseteq_T t \\
t \subseteq_T u & := & \forall x \in t.\ x \in_T u \\
t \in_T u & := & \exists x \in u.\ t =_T u
\end{array}
$$

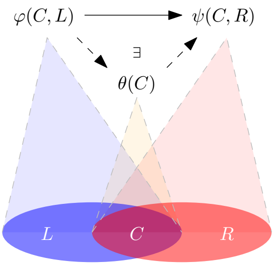
- Allows to suppress the axiom of extensionality
- **No further set-theoretic axioms!**
- Subformula property, for functionality proofs in LJ, sequents have shape

$$\Gamma \vdash t \in_T u \qquad \text{or} \qquad \Gamma \vdash t \subseteq_T u \qquad \text{or} \qquad \Gamma \vdash t =_T u$$

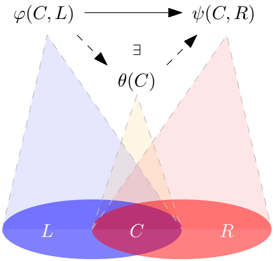
Extraction of terms from proofs



Inspired by **interpolation**
Suppose $\Gamma(c,\vec{l}), \Delta(c,\vec{r}) \vdash l \,\square\, r$.



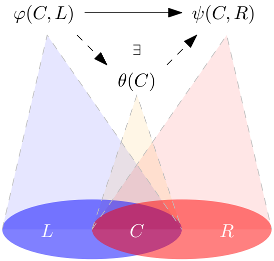
Inspired by **interpolation**
Suppose $\Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \,\square\, r$.
Then we can compute $E(c)$ in NRC such that



Inspired by **interpolation**
Suppose $\Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \,\square\, r$.
Then we can compute $E(c)$ in NRC such that

Inductive invariant

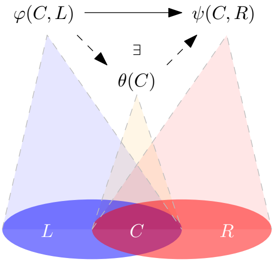
- if $\square$ is $=_T$, then $\Gamma, \Delta \models l = E \wedge r = E$



Inspired by **interpolation**
Suppose $\Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \,\square\, r$.
Then we can compute $E(c)$ in NRC such that

Inductive invariant

- if $\square$ is $=_T$, then $\Gamma, \Delta \models l = E \wedge r = E$
- if $\square$ is $\subseteq_T$, then $\Gamma, \Delta \models l \subseteq E \wedge E \subseteq r$

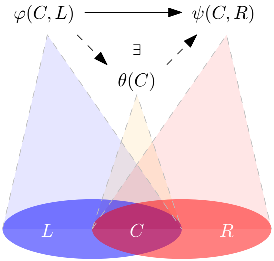


Inspired by **interpolation**
Suppose $\Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \,\square\, r$.
Then we can compute $E(c)$ in NRC such that

Inductive invariant

- if $\square$ is $=_T$, then $\Gamma, \Delta \models l = E \wedge r = E$
- if $\square$ is $\subseteq_T$, then $\Gamma, \Delta \models l \subseteq E \wedge E \subseteq r$
- if $\square$ is $\in_T$, then $\Gamma, \Delta \models l \in E$

Extraction of terms from proofs



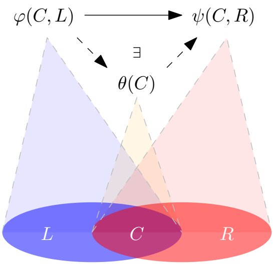
Inspired by **interpolation**
Suppose $\Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \,\square\, r$.
Then we can compute $E(c)$ in NRC such that

Inductive invariant

- if $\square$ is $=_T$, then $\Gamma, \Delta \models l = E \wedge r = E$
- if $\square$ is $\subseteq_T$, then $\Gamma, \Delta \models l \subseteq E \wedge E \subseteq r$
- if $\square$ is $\in_T$, then $\Gamma, \Delta \models l \in E$

Not quite interpolation RHS depends on l

Extraction of terms from proofs



Inspired by **interpolation**
Suppose $\Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \,\square\, r$.
Then we can compute $E(c)$ in NRC such that

Inductive invariant

- if $\square$ is $=_T$, then $\Gamma, \Delta \models l = E \wedge r = E$
- if $\square$ is $\subseteq_T$, then $\Gamma, \Delta \models l \subseteq E \wedge E \subseteq r$
- if $\square$ is $\in_T$, then $\Gamma, \Delta \models l \in E$

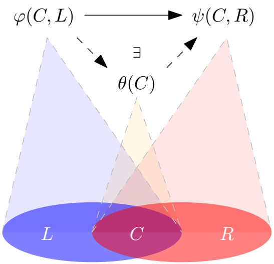
Not quite interpolation RHS depends on l

Going from 3. to 2.

If $\square$ is $\in_T$, then we can compute $E'(c)$ such that

$$\Gamma, \Delta \models l \in E' \wedge E' \subseteq r$$

Extraction of terms from proofs



Inspired by **interpolation**
Suppose $\Gamma(c,\vec{l}), \Delta(c,\vec{r}) \vdash l \,\square\, r$.
Then we can compute $E(c)$ in NRC such that

Inductive invariant

- if $\square$ is $=_T$, then $\Gamma, \Delta \models l = E \wedge r = E$
- if $\square$ is $\subseteq_T$, then $\Gamma, \Delta \models l \subseteq E \wedge E \subseteq r$
- if $\square$ is $\in_T$, then $\Gamma, \Delta \models l \in E$

Not quite interpolation

RHS depends on l

Going from 3. to 2.

If $\square$ is $\in_T$, then we can compute $E'(c)$ such that

$$\Gamma, \Delta \models l \in E' \wedge E' \subseteq r$$

- apply Δ_0 interpolation to $\Gamma \vdash \Delta \Rightarrow l \in_T r$ to obtain $\theta(c, l)$

$\rightsquigarrow$ Γ, Δ jointly imply $l \in \{x \in E \mid \theta(c, l)\} \subseteq r$

Outline

LJ is not complete for functionality proofs wrt classical Tarskian semantics.

$$w \in r;\ \forall x \in l.\ l \in r,\ \forall y \in w.\ l \in r \vdash l \in r$$

Towards classical proofs

LJ is not complete for functionality proofs wrt classical Tarskian semantics.

$$w \in r;\ \forall x \in l.\ l \in r,\ \forall y \in w.\ l \in r \vdash l \in r$$

$\leadsto$ generalize the argument for LK?

While keeping a reasonable algorithmic complexity?

$$\Gamma \vdash t_{\mathbf{1}} \in_{T_{\mathbf{1}}} u_{\mathbf{1}} \ \vee\ \ldots\ \vee\ t_k \in_{T_k} u_k$$

Towards classical proofs

LJ is not complete for functionality proofs wrt classical Tarskian semantics.

$$w \in r;\ \forall x \in l.\ l \in r,\ \forall y \in w.\ l \in r \vdash l \in r$$

$\rightsquigarrow$ generalize the argument for LK?

While keeping a reasonable algorithmic complexity?

$$\Gamma \vdash t_1 \in_{T_1} u_1 \ \vee\ \dots\ \vee\ t_k \in_{T_k} u_k$$

Issues

- What inductive invariant?
- Naive attempts fail because we cannot adapt the above

$$l \in E \qquad \longmapsto \qquad l \in E' \ \wedge\ E' \subseteq r$$

Towards classical proofs

LJ is not complete for functionality proofs wrt classical Tarskian semantics.

$$w \in r;\ \forall x \in l.\ l \in r,\ \forall y \in w.\ l \in r \vdash l \in r$$

$\rightsquigarrow$ generalize the argument for LK?

While keeping a reasonable algorithmic complexity?

$$\Gamma \vdash t_1 \in_{T_1} u_1 \ \vee\ \ldots\ \vee\ t_k \in_{T_k} u_k$$

Issues

- What inductive invariant?
- Naive attempts fail because we cannot adapt the above

$$l \in E \qquad \longmapsto \qquad l \in E' \ \wedge\ E' \subseteq r$$

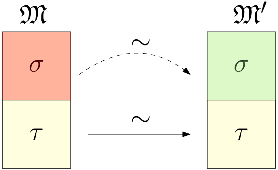
- Unclear how to constructivize the model-theoretic arguments

The model-theoretic argument

- First, an effective correspondence between NRC and interpretations, regarding nested collections as models for $\in$

  interpretations: maps between models defined by FO formulas

- Then, reduction to a model-theoretic result:



Multi-sorted implicit definability

Let Σ be a theory with a multisorted signature $\{\tau, \sigma\}$. Say that σ is implicitly definable from τ when, for every $\mathfrak{M}, \mathfrak{M}' \models \Sigma$ and bijective homomorphism $\mathfrak{M}\big|_\tau \cong \mathfrak{M}'\big|_\tau$, there is a unique extension $\mathfrak{M} \cong \mathfrak{M}'$.

Theorem

If σ is implicitly definable from τ, then there is an interpretation of Σ into $\Sigma\big|_\tau$.

- Is there an effective version?

The inductive invariant, classically

Last slide: deals only with functionality.
What about $\Gamma(c, l), \Delta(c, r) \models l \in r \quad \Longrightarrow \quad \exists E' \; l \in E' \subseteq r$?

The inductive invariant, classically

Last slide: deals only with functionality.
What about $\Gamma(c, l), \Delta(c, r) \models l \in r \quad \Longrightarrow \quad \exists E' \; l \in E' \subseteq r$?

Model-theoretic proof sketch based on a generalization of Beth definability

Generalized Beth definability (Makkai, Chang)

Consider a theory Σ over a single-sorted relational signature $\mathcal{S} \sqcup \{R\}$.
If for every model $\mathfrak{M} = (M, \ldots)$ of Σ, there are $< 2^{|M|}$ bijections $f : M \to M$ such that

- f is an homomorphism over $\mathcal{S}$
- $f(\mathfrak{M}) \models \Sigma$

then there is a parameterized definition φ of R over $\mathcal{S}$:

$$\exists \vec{y}. \forall \vec{x}. R(\vec{x}) \Leftrightarrow \varphi(\vec{x}, \vec{y}) \qquad R \notin FV(\varphi)$$

The inductive invariant, classically

Last slide: deals only with functionality.
What about $\Gamma(c,l), \Delta(c,r) \models l \in r \quad \Longrightarrow \quad \exists E' \; l \in E' \subseteq r$?

Model-theoretic proof sketch based on a generalization of Beth definability

Generalized Beth definability (Makkai, Chang)

Consider a theory Σ over a single-sorted relational signature $\mathcal{S} \sqcup \{R\}$.
If for every model $\mathfrak{M} = (M, \ldots)$ of Σ, there are $< 2^{|M|}$ bijections $f : M \to M$ such that

- f is an homomorphism over $\mathcal{S}$
- $f(\mathfrak{M}) \models \Sigma$

then there is a parameterized definition φ of R over $\mathcal{S}$:

$$\exists \vec{y}. \forall \vec{x}. R(\vec{x}) \Leftrightarrow \varphi(\vec{x}, \vec{y}) \qquad R \notin FV(\varphi)$$

- Non-constructive proof, using saturated models.
- Analogy with Beth: replace "unique" by "few".
- To the best of my knowledge, no proof-theoretic counterpart.

Further work

Besides the aforementioned problems:

- Coq formalization with extraction

  j.w.w. Armaël Guéneau

- Curry-Howard approach to the extraction of NRC terms

  untyped case already implicit in the literature (Sazonov)

- Asymmetric version of the multi-sorted result?

Further work

Besides the aforementioned problems:

- Coq formalization with extraction

  j.w.w. Armaël Guéneau

- Curry-Howard approach to the extraction of NRC terms

  untyped case already implicit in the literature (Sazonov)

- Asymmetric version of the multi-sorted result?

Thanks for listening! Further questions?

Challenges toward an implementation

Effective (polytime) algorithm

Input:

- An implicit definition $\varphi(i, o)$
- An **intuitionistic** (cut-free) proof π of functionality of φ

Output:

- A NRC query $Q(i)$ such that $\varphi(i, o) \Rightarrow Q(i) = o$

1. Code the algorithm?
   - Informal description, no pseudocode
2. Proof object π?
   - Produced by an automated tool — Issue: intuitionistic logic?
   - Produced by the user — Issue: convenient encoding?

Formalization in Coq

Formalize the main statement in an interactive theorem prover

$\exists\pi$ proof of functionality of $\varphi \quad \Rightarrow \quad \exists Q$ NRC expression implementing φ

Formalization in Coq

Formalize the main statement in an interactive theorem prover

$$\exists \pi \text{ proof of functionality of } \varphi \quad \Rightarrow \quad \exists Q \text{ NRC expression implementing } \varphi$$

Requires

Formal definition of Δ_0 formulas, proof derivation, NRC, their semantics

- Inductive families and dependent types
- Bureaucratic paint point: binding construct α-conversion, de Brujin

Proving both interpolation and its higher-order variants

- Literature: only one formalization in Isabelle of interpolation
- Induction with many (bureaucratic) subcases

Formalization in Coq

Formalize the main statement in an interactive theorem prover

$$\exists \pi \text{ proof of functionality of } \varphi \quad \Rightarrow \quad \exists Q \text{ NRC expression implementing } \varphi$$

Requires

Formal definition of Δ_0 formulas, proof derivation, NRC, their semantics

- Inductive families and dependent types
- Bureaucratic paint point: binding construct α-conversion, de Brujin

Proving both interpolation and its higher-order variants

- Literature: only one formalization in Isabelle of interpolation
- Induction with many (bureaucratic) subcases

Benefits of formalizing in Coq

Implementation: proving $\equiv$ implementing the algorithm
Safety: guarantee that the resulting implementation is bug-free

Encoding of proof objects

Recall that an input is a formula $\varphi(i, o)$ and a *proof*

```
Inductive ded {X : Type} {Σ : X -> sort} (θ : memctx Σ) : list (formula Σ) -> formula Σ -> Type :=
| ded_botl : forall Γ Γ' τ (t : tm Σ τ) u,
    [ θ ; Γ ,,, ⊥ ,, Γ' ⊢ t ∈ u ]
          ... (10 subcases omitted)
| ded_eqsetr : forall Γ τ (q q' : tm Σ (β τ)),
    [ θ ; Γ ⊢ q ⊂ q' ] ->
    [ θ ; Γ ⊢ q' ⊂ q ] ->
    [ θ ; Γ ⊢ q ≡ q' ]
| ded_andl : forall Γ Γ' φ ψ τ (t : tm Σ τ) u,
    [ θ ; Γ ,,, φ ,, ψ ,, Γ' ⊢ t ∈ u ] ->
    [ θ ; Γ ,,, φ ∧ ψ ,, Γ' ⊢ t ∈ u ]
| ded_orl : forall Γ Γ' φ ψ τ (t : tm Σ τ) u,
    [ θ ; Γ ,,, φ ,, Γ' ⊢ t ∈ u ] ->
    [ θ ; Γ ,,, ψ ,, Γ' ⊢ t ∈ u ] ->
    [ θ ; Γ ,,, φ ∨ ψ ,, Γ' ⊢ t ∈ u ]
| ded_alll : forall Γ Γ' σ φ τ (t : tm Σ τ) u (v : tm Σ σ) w,
    tm_isvar w ->
    [∈ θ ⊢ v ∈ w] ->
    [ θ ; Γ ,,, φ $ v ,, Γ' ⊢ t ∈ u ] ->
    [ θ ; Γ,,, ∀ w, φ ,, Γ' ⊢ t ∈ u ]
| ded_exl : forall Γ Γ' σ φ τ (t : tm Σ τ) u (v : tm Σ (β σ)),
    tm_isvar v ->
    [ (#None : tm (Σ & σ) σ) ∈ v ↑ σ ,, θ ↑ σ ; Γ ↑ σ ,,, φ ,, Γ' ↑ σ ⊢ (t ∈ u) ↑ σ ] ->
    [ θ ; Γ ,,, ∃ v, φ ,, Γ' ⊢ t ∈ u ]
where "[ θ ; Γ ⊢ φ ]" := (@ded     θ%memctx Γ%ctx φ%fm) : ded scope.
```

Inductive type of proofs

(deep embedding)

- Strongly typed
- Not human-readable

Inputing proof objects directly $\rightsquigarrow$ inconvenient for users

Building complicated objects/functions/proofs in Coq in an interactive mode

```
Require Import Lia.

Definition archimedian :
               forall n m, m <> 0 -> { k | m * k >= n}.
  intros.
  unshelve eexists.
  + apply (plus 5).

    destruct m;[destruct H; auto|].
    exact n.
  + destruct m; simpl.
    { destruct H; auto. }
    lia.
Defined.

Lemma neq0 : forall m, S m <> 0.
 intro; lia.
Qed.

Definition fun_of_archimedian : nat -> nat -> nat :=
 fun n m =>
     let (a, _) := archimedian n (S m) (neq0 _) in a.
Compute (fun_of_archimedian 5 6).
```

```
1 subgoal (1 unfocused at this level)

n, m : nat
H : m <> 0

========================= (1 / 1)

nat
```

- Easier for complex goals

Building complicated objects/functions/proofs in Coq in an interactive mode

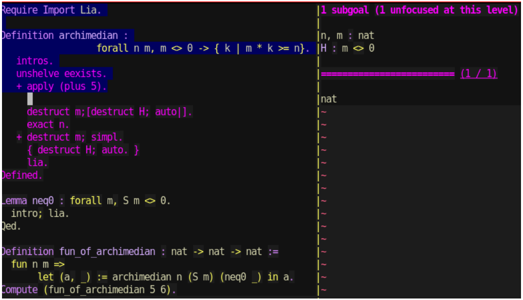
```
Require Import Lia.

Definition archimedian :
                 forall n m, m <> 0 -> { k | m * k >= n}.
   intros.
   unshelve eexists.
   + apply (plus 5).

     destruct m;[destruct H; auto|].
     exact n.
   + destruct m; simpl.
     { destruct H; auto. }
     lia.
Defined.

Lemma neq0 : forall m, S m <> 0.
  intro; lia.
Qed.

Definition fun_of_archimedian : nat -> nat -> nat :=
  fun n m =>
       let (a, _) := archimedian n (S m) (neq0 _) in a.
Compute (fun_of_archimedian 5 6).
```

```
1 subgoal (1 unfocused at this level)

n, m : nat
H : m <> 0

======================== (1 / 1)

nat
```

- Easier for complex goals
- Still inconvenient here
  - Formalized formal proof $\neq$ formal proof
  - Exposes de Brujin notation to users

Tactics

Building complicated objects/functions/proofs in Coq in an interactive mode



- Easier for complex goals
- Still inconvenient here
  - Formalized formal proof $\neq$ formal proof
  - Exposes de Brujin notation to users

Second part of our implementation: special purpose tactics/notations

- Manipulate formulas with actual variables
- $\sim$ small Domain Specific Language

inspired by the Iris proof-mode