Extracting nested relational queries from implicit definitions

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

University of Oxford

November 6th, 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
The nested relational calculus (NRC)

Implicit definition

Implicit to explicit: the nested case

Perspectives
The nested relational calculus (NRC)

Syntax

Types: \[ T, U ::= \emptyset | \text{Set}(T) | 1 | T \times U \]

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

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

- Cartesian structure
- Monad structure on Set
- Idempotent monoid \( \text{Set}(T) \)
- Set difference \( Q \setminus R \)

Generalizes flat relational queries with higher-order types

\[
\text{flat} \cong \text{Set}(\Upsilon^1) \times \ldots \times \text{Set}(\Upsilon^k) \to \text{Set}(\Upsilon^m)
\]
Examples

A flat query

The fiber of a relation $f$ at some point $x$

$$\text{fib} : \mathcal{U} \times \text{Set}(\mathcal{U} \times \mathcal{U}) \rightarrow \text{Set}(\mathcal{U})$$

$$(x, f) \mapsto f^{-1}(x)$$

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

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

A flat query

The fiber of a relation $f$ at some point $x$

$$
\text{fib} : \, U \times \text{Set}(U \times U) \rightarrow \text{Set}(U) \\
(x, f) \quad \mapsto \quad f^{-1}(x)
$$

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

syntactic sugar: case, $=_U$

A genuine nested query

Collect all fibers of $f$

$$
\text{fibs} : \, \text{Set}(U \times U) \rightarrow \text{Set}(U \times \text{Set}(U)) \\
f \quad \mapsto \quad \{(a, f^{-1}(a)) \mid a \in \text{cod}(f)\}
$$

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

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

- maps $\{Q(x) \mid x \in R\}$
- set intersection $Q \cap R$
- case analyses
- basic predicates $=^T : T \times T \rightarrow \text{Bool}$, $\in^T : T \times \text{Set}(T) \rightarrow \text{Bool}$
Expressiveness of NRC

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

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

Proposition

NRC queries $Q(x^T): T \rightarrow \text{Bool}$ correspond exactly to $\Delta_0$ formulas $\varphi(x^T)$. 
Expressiveness of NRC

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

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

Proposition

NRC queries $Q(x^T): T \to \text{Bool}$ correspond exactly to $\Delta_0$ formulas $\varphi(x^T)$. 

$\iff \Delta_0$-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 \text{tt}

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

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

(For the rest of the talk: no finiteness assumptions)
The nested relational calculus (NRC)

Implicit definition

Implicit to explicit: the nested case

Perspectives
### Implicit definability

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

$$\varphi(i, o) \land \varphi(i, o') \Rightarrow o = o'$$

Defines a *partial* function $I \rightarrow O$
Implicit definitions

Implicit definability

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

\[ \varphi(i, o) \land \varphi(i, o') \Rightarrow o = o' \]

 Defines a partial function \( I \rightarrow O \)

Main theorem

Expressible in NRC \( \iff \) Has an implicit definition

- We call a NRC term an explicit definition
- Partial implicit definitions \( \rightarrow \) 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
Main theorem

| Expressible in NRC | ⇔ | 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

$$\text{fibs} : \text{Set}(\mathcal{U} \times \mathcal{U}) \to \text{Set}(\mathcal{U} \times \text{Set}(\mathcal{U}))$$

$$f \mapsto \{(a, f^{-1}(a)) \mid a \in \text{cod}(f)\}$$
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

$$\text{fibs} : \text{Set}(\mathcal{U} \times \mathcal{U}) \rightarrow \text{Set}(\mathcal{U} \times \text{Set}(\mathcal{U}))$$

$$f \quad \mapsto \quad \{(a, f^{-1}(a)) \mid a \in \text{cod}(f)\}$$

- can be converted to an implicit $\varphi(f, F)$
Use-case for implicit→explicit

Main theorem

Expressible in NRC ⇐⇒ 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

\[
\text{fibs} : \text{Set}(\mathcal{U} \times \mathcal{U}) \rightarrow \text{Set}(\mathcal{U} \times \text{Set}(\mathcal{U}))
\]

\[
f \mapsto \{(a, f^{-1}(a)) | a \in \text{cod}(f)\}
\]

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

\[
fibs : \text{Set}(U \times U) \to \text{Set}(U \times \text{Set}(U))
\]

\[
f \mapsto \{(a, f^{-1}(a)) | a \in \text{cod}(f)\}
\]

- can be converted to an implicit $\varphi(f, F)$
- $\varphi(f, F)$ defines a partial function $f \mapsto F$
- $\leadsto$ 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 $\varphi(R) \land \varphi(R') \Rightarrow R \equiv R'$, 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 $\varphi(R) \land \varphi(R') \Rightarrow R \equiv R'$, then there is a FO $\psi(\bar{x})$ 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 $\varphi(R) \land \varphi(R') \Rightarrow R \equiv R'$, then there is a FO $\psi(\overline{x})$ 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 $\varphi(R) \land \varphi(R') \Rightarrow R \equiv R'$, 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 $\theta$ such that

$\varphi \Rightarrow \theta$ and $\theta \Rightarrow \psi$

and $\text{Vocabulary}(\theta) \subseteq \text{Vocabulary}(\varphi) \cap \text{Vocabulary}(\psi)$
Interpolation

The result was already known for the flat case.

**Beth definability**

Let $\varphi(R)$ be a first-order formula. If $\varphi(R) \land \varphi(R') \Rightarrow R \equiv R'$, 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 $\theta$ such that

$\varphi \Rightarrow \theta$ and $\theta \Rightarrow \psi$

and $\text{Vocabulary}(\theta) \subseteq \text{Vocabulary}(\varphi) \cap \text{Vocabulary}(\psi)$

- $\theta$ linear-time computable from a cut-free derivation
- Rather robust result

$\Delta_0$-interpolation, intuitionistic/linear logic...
Fix an implicit definition $\varphi(I, O)$ with $I : \text{Set}(\mathcal{U}^k)$ and $O : \text{Set}(\mathcal{U}^m)$.

Effective proof sketch

Difficulty with the nested case: there is no $M$!
Fix an implicit definition $\varphi(I, O)$ with $I : \text{Set}(\mathcal{U}^k)$ and $O : \text{Set}(\mathcal{U}^m)$.

Effective proof sketch

1. Apply interpolation to 
   $$\varphi(I, O) \land O(\vec{x}) \vdash \varphi(I, O') \Rightarrow O'(\vec{x})$$
   to obtain an explicit $\Delta_0$ definition $\theta(I, \vec{x})$. 
Proof idea for the flat case

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

Effective proof sketch

1. Apply interpolation to

\[
\varphi(I, O) \land O(\vec{x}) \vdash \varphi(I, O') \Rightarrow O'(\vec{x})
\]

to obtain an explicit \( \Delta_0 \) definition \( \theta(I, \vec{x}) \).

2. There is a NRC term \( M : \text{Set}(\mathcal{U}^k) \to \text{Set}(\mathcal{U}^m) \) maximal for \( \subseteq \).
Proof idea for the flat case

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

**Effective proof sketch**

1. Apply interpolation to

   $$\varphi(I, O) \land O(\bar{x}) \vdash \varphi(I, O') \Rightarrow O'(\bar{x})$$

   to obtain an explicit $\Delta_0$ definition $\theta(I, \bar{x})$.

2. There is a NRC term $M : \text{Set}(\mathcal{U}^k) \rightarrow \text{Set}(\mathcal{U}^m)$ maximal for $\subseteq$

   Additionally, $\theta(I, \bar{x}) \iff \theta^M(I, \bar{x})$ for any $\theta \in \Delta_0$. 

Difficulty with the nested case: there is no $M$!
Proof idea for the flat case

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

Effective proof sketch

1. Apply interpolation to $\varphi(I, O) \land O(\vec{x}) \vdash \varphi(I, O') \Rightarrow O'(\vec{x})$
   to obtain an explicit $\Delta_0$ definition $\theta(I, \vec{x})$.

2. There is a NRC term $M : \text{Set}(\mathcal{U}^k) \rightarrow \text{Set}(\mathcal{U}^m)$ maximal for $\subseteq$
   Additionally, $\theta(I, \vec{x}) \iff \theta^M(I, \vec{x})$ for any $\theta \in \Delta_0$.

3. Conclude using $\Delta_0$-comprehension in NRC
   \[ \{ \vec{x} \in M \mid \theta(I, x) \} \]
Fix an implicit definition $\varphi(I, O)$ with $I : \text{Set}(\mathcal{U}^k)$ and $O : \text{Set}(\mathcal{U}^m)$.

**Effective proof sketch**

1. Apply interpolation to
   \[ \varphi(I, O) \land O(\vec{x}) \vdash \varphi(I, O') \Rightarrow O'(\vec{x}) \]
   to obtain an explicit $\Delta_0$ definition $\theta(I, \vec{x})$.

2. There is a NRC term $M : \text{Set}(\mathcal{U}^k) \rightarrow \text{Set}(\mathcal{U}^m)$ maximal for $\subseteq$.
   Additionally, $\theta(I, \vec{x}) \iff \theta^M(I, \vec{x})$ for any $\theta \in \Delta_0$.

3. Conclude using $\Delta_0$-comprehension in NRC
   \[ \{ \vec{x} \in M \mid \theta(I, x) \} \]

Difficulty with the nested case: there is no $M$!
The nested relational calculus (NRC)

Implicit definition

Implicit to explicit: the nested case

Perspectives
### Main theorem

| Expressible in NRC | $\iff$ | Has a $\Delta_0$ implicit definition |

$\impliedby$ automatic translation of implicit definitions to NRC?
Main ineffective result

## Main theorem

| Expressible in NRC | ⇔ | Has a $\Delta_0$ implicit definition |

Does automatic translation of implicit definitions to NRC?

### Problem: a non-constructive proof
- Model-theoretic argument
- A generalization of Beth for multi-sorted structures
Main ineffective result

Main theorem

Expressible in NRC $\iff$ Has a $\Delta_0$ 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 $\Delta_0$ implicit definition
Main effective result

Partial effective result

Expressible in NRC \iff Has an \textit{intuitionistic} $\Delta_0$ implicit definition

Algorithmic content

\textbf{Input:}
- An implicit definition $\phi(i, o)$
- An intuitionistic (cut-free) proof $\pi$ of functionality of $\phi$

\textbf{Output:}
- A NRC query $Q(i)$ such that $\phi(i, o) \Rightarrow Q(i) = o$

\textbf{Caveat: cut-elimination}
Main effective result

Partial effective result

Expressible in NRC $\iff$ Has an intuitionistic $\Delta_0$ implicit definition

Algorithmic content

Input:
- An implicit definition $\varphi(i, o)$
- An intuitionistic (cut-free) proof $\pi$ of functionality of $\varphi$

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 $\Delta_0$ implicit definition

Algorithmic content

Input:
- An implicit definition $\varphi(i, o)$
- An intuitionistic (cut-free) proof $\pi$ of functionality of $\varphi$

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

- Linear-time
- Let’s look at the details…

Caveat: cut-elimination
Δ₀ formulas and intuitionistic sequents

Let’s make several quality-of-life adjustments

\[ t, u ::= x \mid (t, u) \mid \pi_1(t) \mid \pi_2(t) \mid () \]

\[ \varphi, \psi ::= t =_\Sigma u \mid t \neq_\Sigma u \mid \exists x \in t \varphi \mid \forall x \in t \varphi \mid \varphi \land \psi \mid \varphi \lor \psi \]
Let’s make several quality-of-life adjustments

\[
\begin{align*}
  t, u & ::= \ x \ | \ (t, u) \ | \ \pi_1(t) \ | \ \pi_2(t) \ | \ () \\
  \varphi, \psi & ::= \ t =_U u \ | \ t \neq_U u \ | \ \exists x \in t \ \varphi \ | \ \forall x \in t \ \varphi \ | \ \varphi \land \psi \ | \ \varphi \lor \psi
\end{align*}
\]

We use a cut-free version of LJ as our proof system. Cut is admissible
Δ₀ formulas and intuitionistic sequents

Let’s make several quality-of-life adjustments

\[
\begin{align*}
t, u & ::= x \mid (t, u) \mid \pi_1(t) \mid \pi_2(t) \mid () \\
\phi, \psi & ::= t =_\text{Set}(T) u \mid t \neq_\text{Set}(T) u \mid \exists x \in t \phi \mid \forall x \in t \phi \mid \phi \land \psi \mid \phi \lor \psi
\end{align*}
\]

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

**Derived formulas**

\[
\begin{align*}
t =_{\text{Set}(T)} u & ::= t \subseteq_T u \land u \subseteq_T t \\
t \subseteq_T u & ::= \forall x \in t. x \in_T u \\
t \in_T u & ::= \exists x \in u. t =_T u
\end{align*}
\]
Δ₀ formulas and intuitionistic sequents

Let’s make several quality-of-life adjustments

\[ t, u ::= x \mid (t, u) \mid \pi_1(t) \mid \pi_2(t) \mid () \]

\[ \phi, \psi ::= t =_u u \mid t \not= u \mid \exists x \in t \phi \mid \forall x \in t \phi \mid \phi \land \psi \mid \phi \lor \psi \]

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

Derived formulas

\[
\begin{align*}
t =_{\text{Set}(T)} u & ::= t \subseteq_T u \land u \subseteq_T t \\
t \subseteq_T u & ::= \forall x \in t. x \in_T u \\
t \in_T u & ::= \exists x \in u. t =_T u
\end{align*}
\]

- Allows to suppress the axiom of extensionality
- No further set-theoretic axioms!
\( \Delta_0 \) formulas and intuitionistic sequents

Let’s make several quality-of-life adjustments

\[
\begin{align*}
t, u & ::= \ x \mid (t, u) \mid \pi_1(t) \mid \pi_2(t) \mid () \\
\varphi, \psi & ::= \ t =_U u \mid t \neq_U u \mid \exists x \in t \ \varphi \mid \forall x \in t \ \varphi \mid \varphi \land \psi \mid \varphi \lor \psi
\end{align*}
\]

We use a cut-free version of LJ as our proof system. \hspace{1cm} \text{Cut is admissible}

**Derived formulas**

<table>
<thead>
<tr>
<th>Formula</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t =_{\text{Set}(T)} u )</td>
<td>( t \subseteq_T u \land u \subseteq_T t )</td>
</tr>
<tr>
<td>( t \subseteq_T u )</td>
<td>( \forall x \in t. \ x \in_T u )</td>
</tr>
<tr>
<td>( t \in_T u )</td>
<td>( \exists x \in u. \ t =_T u )</td>
</tr>
</tbody>
</table>

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

\[
\Gamma \vdash t \in_T u \quad \text{or} \quad \Gamma \vdash t \subseteq_T u \quad \text{or} \quad \Gamma \vdash t =_T u
\]
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 $\Gamma(c, \bar{l}), \Delta(c, \bar{r}) \vdash l \square r$.
Then we can compute $E(c)$ in NRC such that

**Inductive invariant**

- if $\square$ is $\equiv_T$, then $\Gamma, \Delta \models l = E \land r = E$
Extraction of terms from proofs

Inspired by **interpolation**

Suppose $\Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \sqcap r$.

Then we can compute $E(c)$ in NRC such that

**Inductive invariant**

- if $\sqcap$ is $\sqcap_T$, then $\Gamma, \Delta \models l = E \land r = E$
- if $\sqcap$ is $\subseteq_T$, then $\Gamma, \Delta \models l \subseteq E \land E \subseteq r$
Inspired by **interpolation**

Suppose $\Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \sqcap r$.

Then we can compute $E(c)$ in NRC such that

**Inductive invariant**

- if $\square$ is $=_{T}$, then $\Gamma, \Delta \models l = E \land r = E$
- if $\square$ is $\subseteq_{T}$, then $\Gamma, \Delta \models l \subseteq E \land E \subseteq r$
- 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 $\Gamma, \Delta \models l = E \land r = E$
- if $\square$ is $\subseteq_T$, then $\Gamma, \Delta \models l \subseteq E \land E \subseteq r$
- 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 $\Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \sqcap r$.

Then we can compute $E(c)$ in NRC such that

**Inductive invariant**

- if $\square$ is $=_{T}$, then $\Gamma, \Delta \models l = E \land r = E$
- if $\square$ is $\subseteq_{T}$, then $\Gamma, \Delta \models l \subseteq E \land E \subseteq r$
- 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

$$\Gamma, \Delta \models l \in E' \land E' \subseteq r$$
Inspired by **interpolation**

Suppose \( \Gamma(c, \vec{l}), \Delta(c, \vec{r}) \vdash l \Box r \).

Then we can compute \( E(c) \) in NRC such that

**Inductive invariant**

- if \( \Box \) is \( =_T \), then \( \Gamma, \Delta \models l = E \land r = E \)
- if \( \Box \) is \( \subseteq_T \), then \( \Gamma, \Delta \models l \subseteq E \land E \subseteq r \)
- if \( \Box \) is \( \in_T \), then \( \Gamma, \Delta \models l \in E \)

Not quite interpolation  
RHS depends on \( l \)

**Going from 3. to 2.**

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

\[
\Gamma, \Delta \models l \in E' \land E' \subseteq r
\]

- apply \( \Delta_0 \) interpolation to \( \Gamma \vdash \Delta \Rightarrow l \in_T r \) to obtain \( \theta(c, l) \)

\[
\leadsto \Gamma, \Delta \text{ jointly imply } l \in \{ x \in E \mid \theta(c, l) \} \subseteq r
\]
The nested relational calculus (NRC)

Implicit definition

Implicit to explicit: the nested case

Perspectives
Towards classical proofs

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

\[ w \in r; \forall x \in l. l \in r, \forall y \in w. l \in r \vdash l \in r \]
LJ is not complete for functionality proofs w.r.t. classical Tarskian semantics.

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

\[ \vdash \text{generalize the argument for LK?} \]

While keeping a reasonable algorithmic complexity?

\[ \Gamma \vdash t_1 \in T_1 \ u_1 \ \lor \ldots \ \lor \ t_k \in T_k \ u_k \]
Towards classical proofs

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

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

\[ \leadsto \text{generalize the argument for LK?} \]

While keeping a reasonable algorithmic complexity?

\[ \Gamma \vdash t_1 \in T_1 \ u_1 \ \lor \ldots \ \lor t_k \in T_k \ u_k \]

Issues

▶ What inductive invariant?

▶ Naive attempts fail because we cannot adapt the above

\[ l \in E \quad \leadsto \quad l \in E' \ \land \ E' \subseteq r \]
Towards classical proofs

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

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

\[ \leadsto \text{generalize the argument for LK?} \]

While keeping a reasonable algorithmic complexity?

\[ \Gamma \vdash t_1 \in T_1 \ u_1 \lor \ldots \lor t_k \in T_k \ u_k \]

Issues

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

\[ l \in E \quad \iff \quad l \in E' \land E' \subseteq r \]

- 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 ∈ 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 \{τ, σ\}. Say that σ is implicitly definable from τ when, for every \(M, M' \models Σ\) and bijective homomorphism \(M|_τ \cong M'|_τ\), there is a unique extension \(M \cong M'\).

Theorem
If σ is implicitly definable from τ, then there is an interpretation of Σ into \(Σ|_τ\).

Is there an effective version?
The inductive invariant, classically

Last slide: deals only with functionality.
What about $\Gamma(c, l), \Delta(c, r) \models l \in r \implies \exists E' \ l \in E' \subseteq r$?
The inductive invariant, classically

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

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

**Generalized Beth definability (Makkai, Chang)**

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

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

then there is a parameterized definition $\varphi$ of $R$ over $S$: 

$$\exists \bar{y}. \forall \bar{x}. R(\bar{x}) \iff \varphi(\bar{x}, \bar{y}) \quad R \notin \text{FV}(\varphi)$$
The inductive invariant, classically

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

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

**Generalized Beth definability (Makkai, Chang)**

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

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

then there is a parameterized definition \( \varphi \) of \( R \) over \( S \):

\[
\exists \vec{y}. \forall \vec{x}. R(\vec{x}) \iff \varphi(\vec{x}, \vec{y}) \quad R \notin FV(\varphi)
\]

- 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?
Besides the aforementioned problems:

- Coq formalization with extraction

- 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 $\pi$ of functionality of $\varphi$

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 $\pi$?
   - Produced by an automated tool  
     Issue: intuitionistic logic?
   - Produced by the user  
     Issue: convenient encoding?
Formalize the main statement in an interactive theorem prover

\[ \exists \pi \text{ proof of functionality of } \varphi \Rightarrow \exists Q \text{ NRC expression implementing } \varphi \]
Formalization in Coq

Formalize the main statement in an interactive theorem prover

\[ \exists \pi \text{ proof of functionality of } \varphi \implies \exists Q \text{ NRC expression implementing } \varphi \]

**Requires**

- Formal definition of \( \Delta_0 \) formulas, proof derivation, NRC, their semantics
  - Inductive families and dependent types
  - Bureaucratic paint point: binding construct
    - \( \alpha \)-conversion, de Bruijn
- 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

\[ \exists \pi \text{ proof of functionality of } \varphi \implies \exists Q \text{ NRC expression implementing } \varphi \]

Requires

Formal definition of $\Delta_0$ formulas, proof derivation, NRC, their semantics

- Inductive families and dependent types
- Bureaucratic paint point: binding construct
  \[ \alpha \text{-conversion, de Bruijn} \]

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 type of proofs
(deep embedding)

- Strongly typed
- Not human-readable

Inputing proof objects directly $\leadsto$ inconvenient for users
Building complicated objects/functions/proofs in Coq in an interactive mode

Easier for complex goals
Building complicated objects/functions/proofs in Coq in an interactive mode

Easier for complex goals

Still inconvenient here
  ▶ Formalized formal proof ≠ 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 Bruijn notation to users

Second part of our implementation: special purpose tactics/notations
- Manipulate formulas with actual variables
- small Domain Specific Language inspired by the Iris proof-mode