An Implementation of Set Theory with Pointed Graphs in Dedukti

Valentin Blot Gilles Dowek Thomas Traversié

LFMTP 2022

What is DEDUKTI?

- Many provers: COQ, HOL LIGHT, PVS

  $\hookrightarrow$ Interoperability between theorem provers

- DEDUKTI:

  – Type-checker for the $\lambda\Pi$-*calculus modulo theory*: λ-calculus with dependent types and rewriting rules

  – Logical framework: many theories can be expressed

  $\hookrightarrow$ Aim: universal

Why set theory?

- Standard theory
  - 'Paradise' for mathematicians (Hilbert)
  - Used in several theorem provers: MIZAR, ISABELLE/ZF, ATELIER B
- Implementation in DEDUKTI
  - State each axiom
  - Define each axiom as a rewriting rule [Crabbé, 1974]
  - Encode sets using *pointed graphs* [Dowek-Miquel, 2007]

Overview of the different theories

Name of the theory	**Set theory**	**Set theory with pointed graphs – IZmod**	**Set theory implemented in Dedukti**
Formalization	Predicate logic	*Deduction modulo theory*	$\lambda\Pi$-*calculus modulo theory*
Reference	J.-L. KRIVINE, *Introduction to Axiomatic Set Theory*, 1983	G. DOWEK, A. MIQUEL, *Cut elimination for Zermelo set theory*, 2007	

Contributions of this paper

- Adapt the encoding with pointed graphs from *Deduction modulo theory* to $\lambda\Pi$-*calculus modulo theory*

- Implement it in DEDUKTI
  $\hookrightarrow$ Define an inductive sort of formulas

Outline

IZmod theory

Implementation of IZmod in DEDUKTI

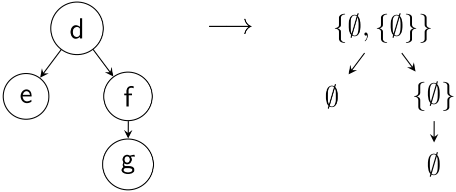
Part 1:

IZmod theory

Theory of pointed graphs IZmod

- Pointed graph: directed graph with a root [Aczel, 1988]

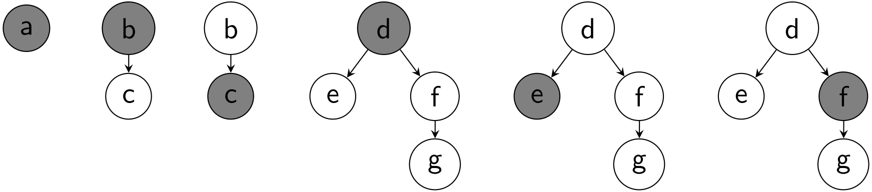
- Interpretation depends on the location of the root



Root at e or g: $\emptyset$
Root at f: $\{\emptyset\}$
Root at d: $\{\emptyset, \{\emptyset\}\}$

Theory of pointed graphs IZmod

- Examples of representation of sets by pointed graphs



$\emptyset$ $\{\emptyset\}$ $\emptyset$ $\{\emptyset, \{\emptyset\}\}$ $\emptyset$ $\{\emptyset\}$

Theory of pointed graphs IZmod

- Definitions

$x \; \eta_a \; y$	edge from y to x in pointed graph a
a/x	changes the root of pointed graph a to be node x
$\mathrm{root}(a)$	returns the root of pointed graph a

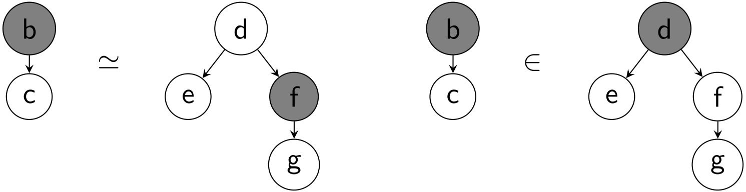
- Rewriting rules

$$x \; \eta_{a/z} \; y \longrightarrow x \; \eta_a \; y$$

$$\mathrm{root}(a/x) \longrightarrow x$$

$$(a/x)/y \longrightarrow a/y$$

Relations between pointed graphs



- Bisimilarity

$$
\begin{aligned}
a \simeq b \longrightarrow\ & \exists r,\ r\ root(a)\ root(b) \\
& \wedge\ \forall x \forall x'\ \forall y\ (x'\ \eta_a\ x\ \wedge\ r\ x\ y\ \Rightarrow\ \exists y'\ (y'\ \eta_b\ y\ \wedge\ r\ x'\ y')) \\
& \wedge\ \forall y \forall y'\ \forall x\ (y'\ \eta_b\ y\ \wedge\ r\ x\ y\ \Rightarrow\ \exists x'\ (x'\ \eta_a\ x\ \wedge\ r\ x'\ y'))
\end{aligned}
$$

- Membership relation

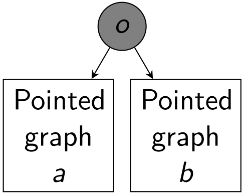
$$a \in b \longrightarrow \exists x\ (x\ \eta_b\ root(b)\ \wedge\ a \simeq (b/x))$$

Constructions

- For each axiom of set theory, we need a constructor defined by rewriting rules

- Pairing: $\forall a \forall b \exists c \forall x \; (x \in c \Leftrightarrow (x \simeq a \vee x \simeq b))$

Creation of $c = \{a, b\}$



Nodes of $a \neq$ nodes of $b \neq o$

Constructions

- Disjoint injections i, j such that o is not in their images

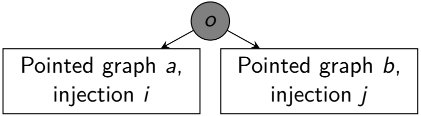
$$
\begin{array}{llll}
i'(i(x)) \longrightarrow x & I(i(x)) \longrightarrow \top & I(j(x)) \longrightarrow \bot & I(o) \longrightarrow \bot \\
j'(j(x)) \longrightarrow x & J(j(x)) \longrightarrow \top & J(i(x)) \longrightarrow \bot & J(o) \longrightarrow \bot
\end{array}
$$

with inverses i', j' and images I, J

Constructions

- Pairing: $\quad \mathrm{root}(\{a, b\}) \longrightarrow o$

$$
\begin{aligned}
x\ \eta_{\{a,b\}}\ x' \longrightarrow\ & (\exists y \exists y'\ (x = i(y) \wedge x' = i(y') \wedge y\ \eta_a\ y')) \\
\vee\ & (\exists y \exists y'\ (x = j(y) \wedge x' = j(y') \wedge y\ \eta_b\ y')) \\
\vee\ & (x = i(\mathrm{root}(a)) \wedge x' = o) \\
\vee\ & (x = j(\mathrm{root}(b)) \wedge x' = o)
\end{aligned}
$$



- Similar constructions for the other axioms

Set theories

Z

Pairing
Union
Extensionality
Powerset
Comprehension
Infinity

IZst

Strong Extensionality
Transitive Closure

ZF

Foundation
Replacement

Set theories

- Strong Extensionality axiom

$$
\begin{aligned}
\forall x_1 \ldots \forall x_n \forall a \forall b\ (R(a, b) \\
&\wedge\ \forall x \forall x' \forall y\ (x' \in x \wedge R(x, y) \Rightarrow \exists y'\ (y' \in y \wedge R(x', y'))) \\
&\wedge\ \forall y \forall y' \forall x\ (y' \in y \wedge R(x, y) \Rightarrow \exists x'\ (x' \in x \wedge R(x', y'))) \\
&\Rightarrow a \simeq b)
\end{aligned}
$$

  where $R(a, b)$ is a formula with free variables $x_1, \ldots, x_n$

- Transitive Closure axiom

$$\forall a \exists e\ (a \subseteq e \wedge \underbrace{\forall x \forall y\ (x \in y \wedge y \in e \Rightarrow x \in e)}_{\text{transitive set}})$$

IZmod and IZst

- The theory of pointed graphs IZmod validates IZst [pen and paper proof, Dowek-Miquel, 2007]

- Each axiom of IZst is a lemma in IZmod
  $+$ Intermediary lemmas on the structure of pointed graphs
  $=$ 53 lemmas necessary

  Pairing (lemma 43): $\forall x\ (x \in \{a, b\} \Leftrightarrow (x \simeq a \vee x \simeq b))$
  $\hookleftarrow$ Lemma 36: $(\{a, b\}/i(\mathrm{root}(a))) \simeq a$
  $\hookleftarrow$ Lemma 37: $(\{a, b\}/j(\mathrm{root}(b))) \simeq b$

- Implementation in Dedukti: formal proofs of the 53 lemmas

Part 2:

Implementation of IZmod in DEDUKTI

Implementation in Dedukti

- Universe of sorts $\mathit{Set} : \mathtt{TYPE}$ [Blanqui *et al.*, 2021]
  Function $\mathit{El} : \mathit{Set} \to \mathtt{TYPE}$

```
constant symbol graph : Set;
constant symbol node : Set;
```

- Simplification in $\lambda\Pi$-*calculus modulo theory*: sorts defined via *El node* and *El prop*
  - Sort of classes: $\mathit{El\ node} \to \mathit{El\ prop}$
  - Sort of binary relations: $\mathit{El\ node} \to \mathit{El\ node} \to \mathit{El\ prop}$

Implementation in Dedukti

- Signature in DEDUKTI: 28 symbols

```
symbol eta : El graph → El node → El node → El prop;
symbol root : El graph → El node;
// change of root
symbol cr : El graph → El node → El graph;
```

+ Injections (o, `i`, `j`, `i'`, `j'`, ρ, ρ`'`) + Natural numbers by nodes + Relations (`simeq`, $\in$) + Constructions (`pair`, `join`, `powerset`, `omega`, `closure`)

- In $\lambda\Pi$-*calculus modulo theory*, no need for some symbols:
  $rel(x, y, r) \rightarrow r\ x\ y$, $mem(P, x) \rightarrow P\ x$, ...

Implementation in Dedukti

- Rewriting rules

```
rule eta (cr $a $z) $x $y ↪ eta $a $x $y;
rule root (cr $a $x) ↪ $x;
rule (cr (cr $a $x) $y) ↪ cr $a $y;
```

- Most of the 53 lemmas are provable without much modification

From a footnote to formulas

- Comprehension axiom schema in IZmod:

$$\forall b \; \exists \; comp_{x,P}(b) \; \forall a \; [a \in comp_{x,P}(b) \Leftrightarrow (a \in b \; \wedge \; P(x \longleftarrow a))] \; (*)$$

- Domain of propositions is restricted

(*) P formula in the language $\{\simeq, \; \in\}$ and with quantifiers on pointed graphs

- Deep embedding for this class of formulas

Class of formulas $\xrightarrow{interpretation}$ class of propositions

The language of formulas

- Sort of formulas

```
constant symbol formula : Set;
```

- Logics on formulas

```
constant symbol eqF : El nat → El nat → El formula;
constant symbol inF : El nat → El nat → El formula;
```

Same for `andF`, `orF`, `impF`, `allF`, `exF`, `fF`, `tF`

The language of formulas

- Restricted class of formulas defined by induction

```
constant symbol recF : Π (P : El formula → El prop),
π(‘∀ x, ‘∀ y, P (eqF x y))
→ π(‘∀ x, ‘∀  y, P (inF x y))
→ π(‘∀ f, ‘∀ g, (P f ∧ P g) ⇒ (P (andF f g)))
→ π(‘∀ f, ‘∀ g, (P f ∧ P g) ⇒ (P (orF f g)))
→ π(‘∀ f, ‘∀ g, (P f ∧ P g) ⇒ (P (impF f g)))
→ π(‘∀ f, (P f) ⇒ (‘∀ x, P (allF x f)))
→ π(‘∀ f, (P f) ⇒ (‘∀ x, P (exF x f)))
→ π(P tF)
→ π(P fF)
→ π(‘∀ f, P f);
```

Valuation

- Valuation: `El nat → El graph`

- Substitution: $\sigma,\ x \leftarrow a$

```
symbol update : (El nat → El graph) → El nat
        → El graph → (El nat → El graph)
```

$$(update\ \sigma\ x\ a)\ y = \begin{cases} \sigma\ y & \text{if } y \neq x \\ a & \text{if y = x} \end{cases}$$

$\hookrightarrow$ Need to manage a decision procedure in a rewriting rule:

to check if $x = y$, we successively decrement them to reach 0

Interpretation of formulas

- Interpretation into IZmod

```
symbol interpretation : (El nat → El graph)
            → El formula → El prop;
```

- Rewriting rules

$$[\![andF\ f\ g]\!]_\sigma \longrightarrow [\![f]\!]_\sigma \wedge [\![g]\!]_\sigma$$

$$[\![orF\ f\ g]\!]_\sigma \longrightarrow [\![f]\!]_\sigma \vee [\![g]\!]_\sigma$$

$$[\![impF\ f\ g]\!]_\sigma \longrightarrow [\![f]\!]_\sigma \Rightarrow [\![g]\!]_\sigma$$

$$[\![tF]\!]_\sigma \longrightarrow \top$$

$$[\![fF]\!]_\sigma \longrightarrow \bot$$

Interpretation of formulas

- Rewriting rules

$$[\![eqF\ x\ y]\!]_\sigma \longrightarrow (\sigma\ x) \simeq (\sigma\ y)$$

$$[\![inF\ x\ y]\!]_\sigma \longrightarrow (\sigma\ x) \in (\sigma\ y)$$

$$[\![allF\ x\ f]\!]_\sigma \longrightarrow \forall a\ [\![f]\!]_{\sigma, x \leftarrow a}$$

$$[\![exF\ x\ f]\!]_\sigma \longrightarrow \exists a\ [\![f]\!]_{\sigma, x \leftarrow a}$$

- Class of formulas $\xrightarrow{interpretation}$ class of propositions

Using the language of formulas

- Comprehension constructor

$$a \in comp_{x,P}(b) \Leftrightarrow [a \in b \ \wedge \ P(x \longleftarrow a)]$$

```
symbol comp : El graph → (El nat → El graph)
      → El formula → El graph;
```

- Empty set

```
rule empty_set ↪ comp omega (λ _, omega) fF;
```

where `omega` is the pointed graph of von Neumann ordinals

Using the language of formulas

- Inductive set c: $\emptyset \in c \wedge (a \in c \Rightarrow a \cup \{a\} \in c)$

  ```
  rule Ind $c ↪ (empty_set ∈ $c)
  ∧ (`∀ a, (a ∈ $c) ⇒ ((join (pair a (pair a a))) ∈ $c));
  ```

  where $\cup\{a, \{a, a\}\} = \cup\{a, \{a\}\} = a \cup \{a\}$

- Axiom of infinity: $Ind(\texttt{omega})$

- Proof of the remaining lemmas

Conclusion

- Formal proof that IZmod validates IZst
- Implementation of set theory in DEDUKTI with an encoding of sets and an inductive sort of formulas
- Future work:
  - Normalization property
  - Translation of proofs between ISABELLE/ZF and DEDUKTI