From Sets to Types

Using the B Method with modern proof tools

Vincent Trélat

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

Introduction

B core language

Core (mathematical) language:

- **set theory** (ZF-like)
- axiom of **choice** (`CHOICE`)
- native numerals ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$)
- **well-definedness** on terms (partial operators)

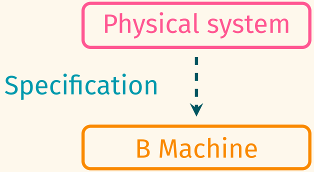
Introduction

B method

Physical system

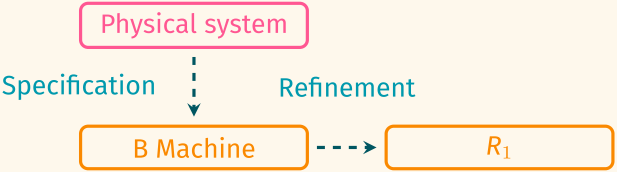
Introduction

B method



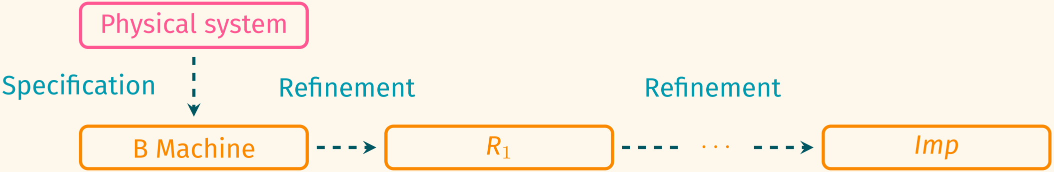
Introduction

B method



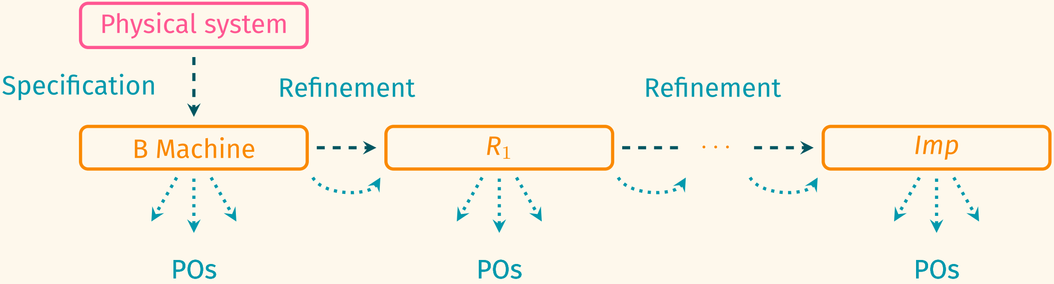
Introduction

B method



Introduction

B method



Proof Obligations

Proof Obligations

automated

interactive

Proof Obligations

mp, pp, rp

ppTrans + SMT solvers

automated

ATELIER B

interactive

Introduction

B method

Proof Obligations

mp, pp, rp

ppTrans + SMT solvers

automated

+ CVC5

BEer

atelier B

interactive

BARReL

Introduction

Because Germany taught me to lean on beer,

Introduction

Because Germany taught me to lean on beer,

are written in LEAN

Introduction

SMT-LIB (up to v2.6)

- Standard input format for SMT solvers (e.g. z3, cvc5, veriT)
- Based on many-sorted **first-order logic**
- Comes with many theories (e.g. arrays, integer and real arithmetic)

Introduction

SMT-LIB (up to v2.6)

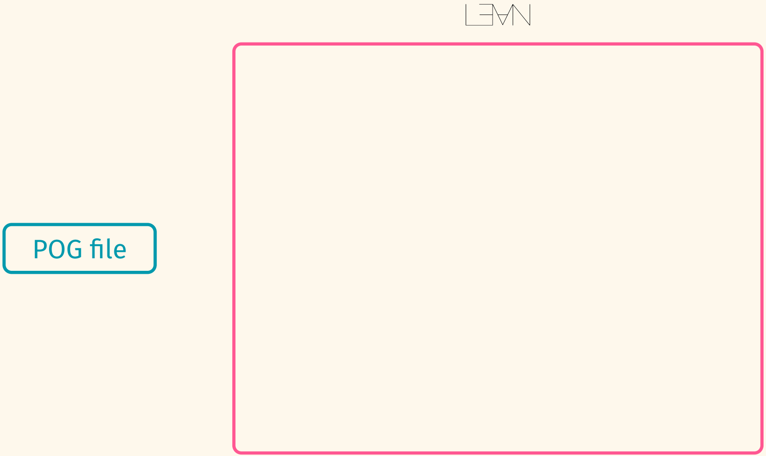
- Standard input format for SMT solvers (e.g. z3, cvc5, veriT)
- Based on many-sorted **first-order logic**
- Comes with many theories (e.g. arrays, integer and real arithmetic)

SMT-LIB v2.7

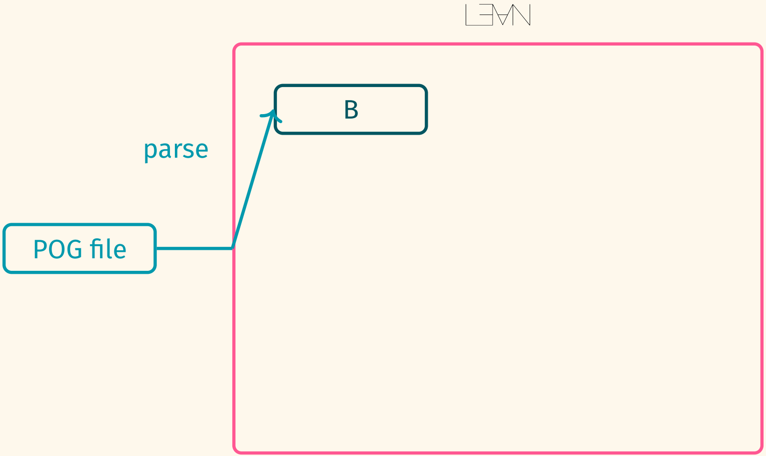
- Brings **higher-order constructs** through λ-abstractions
- Brings **higher-order types** through arrow type constructor
- Only supported by cvc5 yet

Architecture of

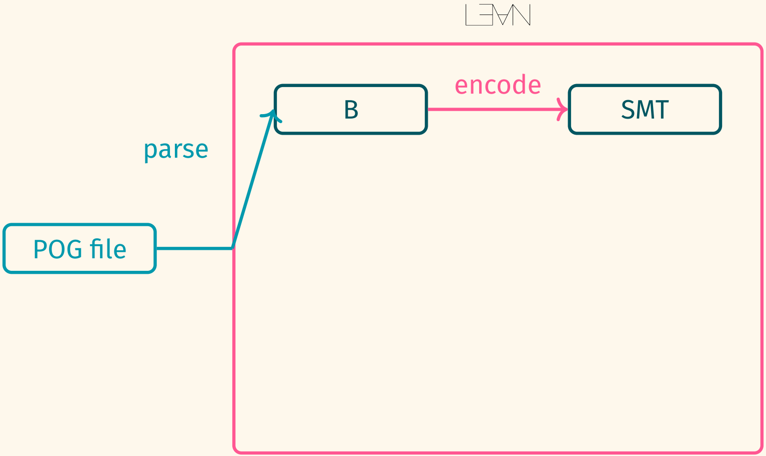
Architecture of



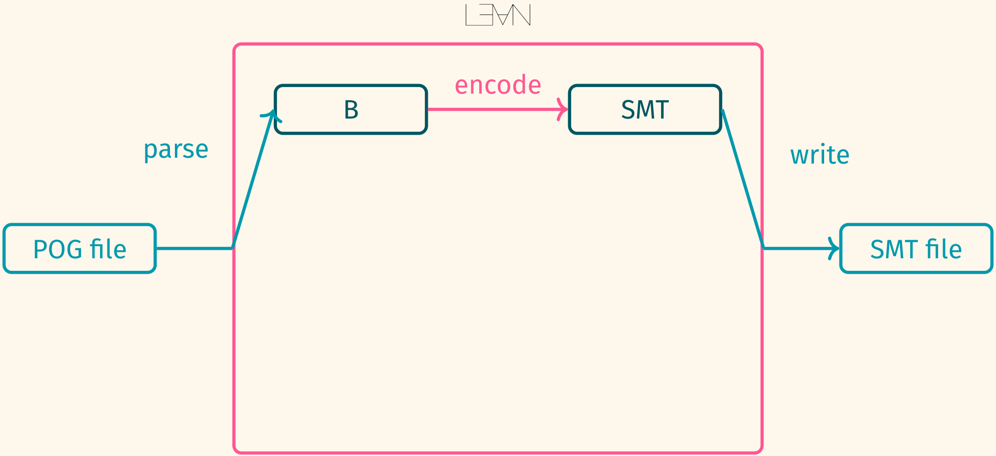
Architecture of 🍺



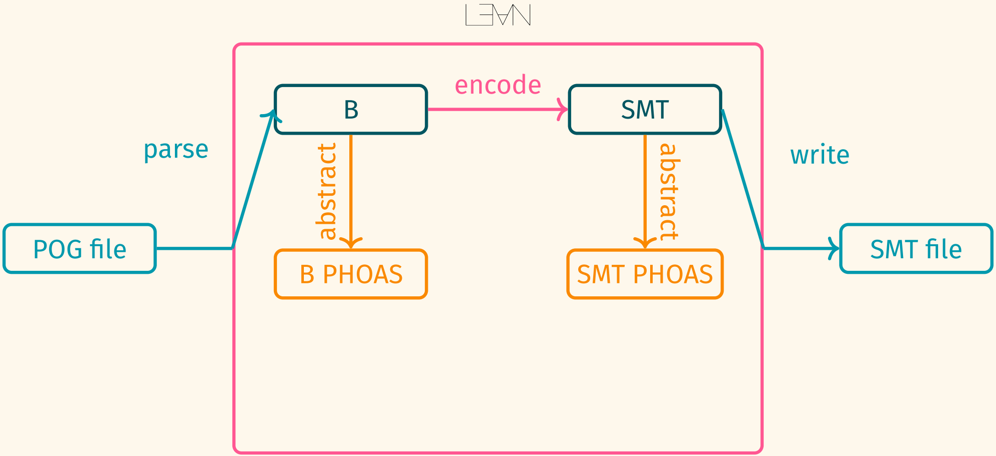
Architecture of 🍺



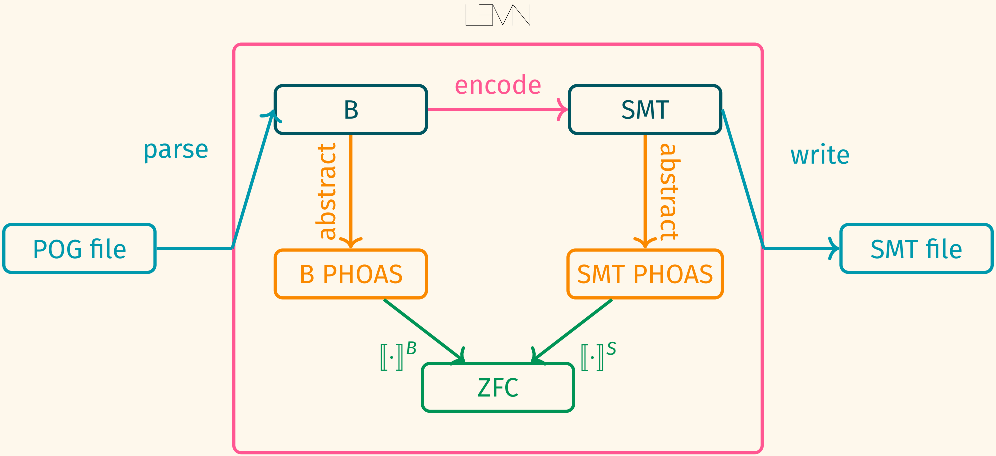
Architecture of



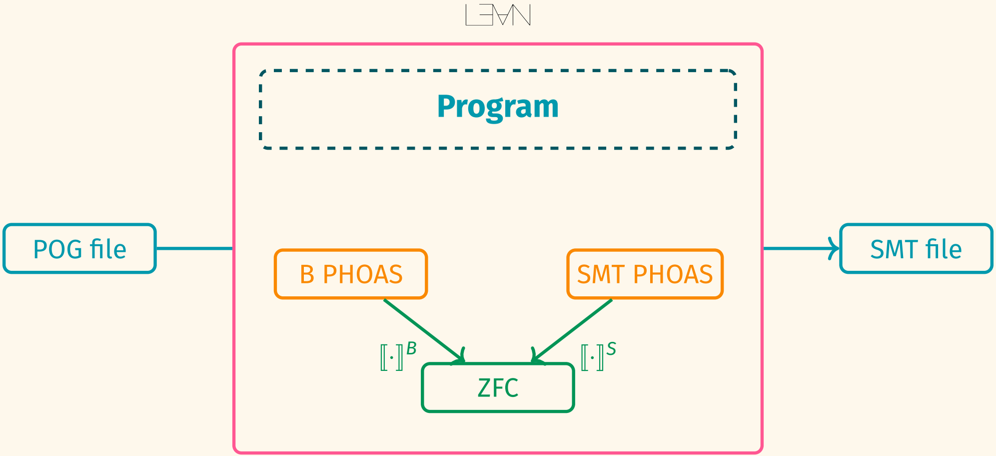
Architecture of 🍺



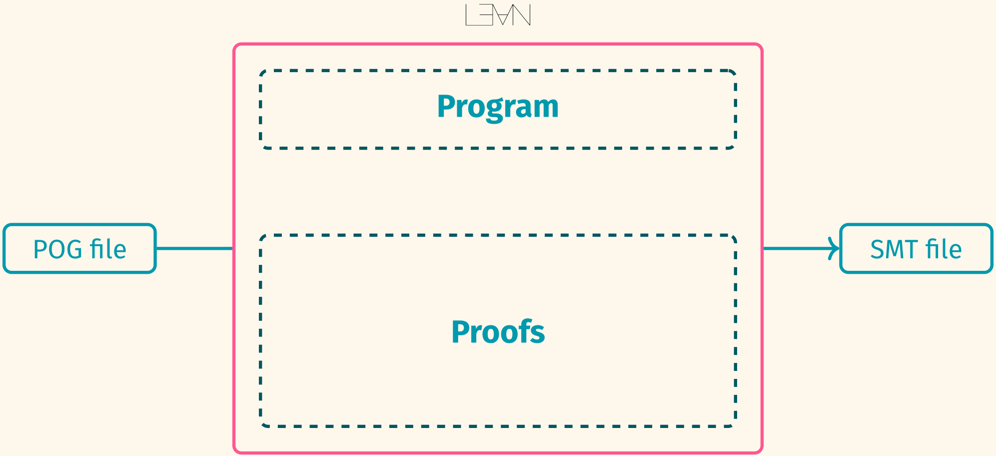
Architecture of 🍺



Architecture of 🍺



Architecture of 🍺



Overview

FO encoding

HO encoding 🍺

Overview

FO encoding

- FOL

HO encoding 🍺

- HOL

Overview

FO encoding

- FOL
- Specification of sets via $\in$, P and C

HO encoding 🍺

- HOL
- Definition of sets via characteristic predicates

Overview

SETS

```
  S = {e1, e2, e3}
```

Overview

SETS

```
  S = {e1, e2, e3}
```

FO encoding

```
(declare-sort P 1)
(declare-sort C 2)
(declare-fun S () (P Int))
(declare-fun e1 () Int)
(declare-fun e2 () Int)
(declare-fun e3 () Int)
(assert (distinct e1 e2 e3))
(declare-fun ∈0 ((Int) (P Int)) Bool)

(assert (forall ((x Int)) (=
  (∈0 x S)
  (or (= x e1) (= x e2) (= x e3)))))
```

Overview

SETS

```
S = {e1, e2, e3}
```

FO encoding

```
(declare-sort P 1)
(declare-sort C 2)
(declare-fun S () (P Int))
(declare-fun e1 () Int)
(declare-fun e2 () Int)
(declare-fun e3 () Int)
(assert (distinct e1 e2 e3))
(declare-fun ∈0 ((Int) (P Int)) Bool)

(assert (forall ((x Int)) (=
  (∈0 x S)
  (or (= x e1) (= x e2) (= x e3)))))
```

HO encoding 🍺

```
(declare-const e1 Int)
(declare-const e2 Int)
(declare-const e3 Int)
(assert (distinct e1 e2 e3))
(define-const S (→ Int Bool)
  (lambda ((x Int))
    (or (= x e1) (= x e2) (= x e3))))
```

Overview

FO encoding

- FOL
- Specification of sets via $\in$, P and C

HO encoding 🍺

- HOL
- Definition of sets via characteristic predicates

Overview

FO encoding

- FOL
- Specification of sets via $\in$, P and C
- Only expressions like $x \in S$ are encoded

HO encoding 🍺

- HOL
- Definition of sets via characteristic predicates
- Sets alone make sense; $x \in S$ is true by definition

Overview

FO encoding

- FOL
- Specification of sets via $\in$, P and C
- Only expressions like $x \in S$ are encoded
- Functions are functional relations

HO encoding 🍺

- HOL
- Definition of sets via characteristic predicates
- Sets alone make sense; $x \in S$ is true by definition
- Functions are (sometimes) functions

Suppose we have a function $f \in A \nrightarrow B$.

Suppose we have a function $f \in A \rightarrowtail B$.

FO encoding

f is a relation between A and B:

$$f \subseteq A \times B$$

f is functional:

$$\forall x\, y\, z,\ x \mapsto y \in f \ \wedge\ x \mapsto z \in f \Rightarrow y = z$$

Suppose we have a function $f \in A \nrightarrow B$.

FO encoding

f is a relation between A and B:

$$f \subseteq A \times B$$

f is functional:

$$\forall x\, y\, z,\ x \mapsto y \in f \ \land\ x \mapsto z \in f \Rightarrow y = z$$

HO encoding 🍺

f is a total function from A to $B \uplus \{\star\}$:

$$f \in (B \uplus \{\star\})^A$$

Suppose we have a function $f \in A \nrightarrow B$.

FO encoding

f is a relation between A and B:

$$f \subseteq A \times B$$

f is functional:

$$\forall x\, y\, z,\ x \mapsto y \in f \ \wedge\ x \mapsto z \in f \Rightarrow y = z$$

HO encoding 🍺

f is a total function from A to $B \uplus \{\star\}$:

$$f \in (B \uplus \{\star\})^A$$

```
(declare-datatype Option
  (par (T) ((some (the T)) (none))))
```

Suppose we have a function $f \in A \rightarrow\!\!\!\!\!+ B$. Let τ_A and τ_B represent the types of A and B respectively.

Suppose we have a function $f \in A \nrightarrow B$. Let τ_A and τ_B represent the types of A and B respectively.

FO encoding

```
(declare-sort P 1)
(declare-sort C 2)
(declare-const f (P (C τ_A τ_B)))
(declare-fun
  ∈_0 (τ_A τ_B (P (C τ_A τ_B))) Bool)
(assert
  (forall ((x τ_A) (y τ_B) (z τ_B))
    (⇒ (and (∈_0 x y f) (∈_0 x z f))
        (= y z))))
```

Suppose we have a function $f \in A \pto B$. Let τ_A and τ_B represent the types of A and B respectively.

FO encoding

```
(declare-sort P 1)
(declare-sort C 2)
(declare-const f (P (C τ_A τ_B)))
(declare-fun
  ∈₀ (τ_A τ_B (P (C τ_A τ_B))) Bool)
(assert
  (forall ((x τ_A) (y τ_B) (z τ_B))
    (⇒ (and (∈₀ x y f) (∈₀ x z f))
        (= y z))))
```

HO encoding

```
(declare-const f (→ τ_A (Option τ_B)))
```

Suppose we have a function $f \in A \nrightarrow B$. Let τ_A and τ_B represent the types of A and B respectively.

FO encoding

```
(declare-sort P 1)
(declare-sort C 2)
(declare-const f (P (C τ_A τ_B)))
(declare-fun
  ∈₀ (τ_A τ_B (P (C τ_A τ_B))) Bool)
(assert
  (forall ((x τ_A) (y τ_B) (z τ_B))
    (⇒ (and (∈₀ x y f) (∈₀ x z f))
        (= y z))))
```

HO encoding

```
(declare-const f (→ τ_A (Option τ_B)))
```

+ specification that **dom** $f \subseteq A$ and **ran** $f \subseteq B$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$\forall a : \texttt{int} \cdot \exists b : \texttt{int}, f : \texttt{set}(\tau \times \texttt{int}) \cdot f \in S \rightarrowtail a..b$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$\forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}(\tau \times \mathtt{int}) \cdot f \in S \rightarrowtail a..b$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$\forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}\,(\tau \times \mathtt{int}) \cdot f \in S \nrightarrow a..b \wedge S \subseteq \mathbf{dom}(f) \wedge \_\mathtt{inj}(f)$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$\forall a : \texttt{int} \cdot \exists b : \texttt{int}, f : \texttt{set}(\tau \times \texttt{int}) \cdot f \in S \nrightarrow a..b \wedge S \subseteq \mathbf{dom}(f) \wedge \_\texttt{inj}(f)$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$\begin{aligned}\forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}\,(\tau \times \mathtt{int}) \cdot \\ & f \in S \leftrightarrow a..b & \wedge \\ & \_\mathtt{func}(f) & \wedge \\ & S \subseteq \mathbf{dom}(f) & \wedge \\ & \_\mathtt{inj}(f)\end{aligned}$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$\begin{aligned} \forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}\,(\tau \times \mathtt{int}) \cdot \\ & f \in S \leftrightarrow a..b && \wedge \\ & \mathtt{\_func}(f) && \wedge \\ & S \subseteq \mathbf{dom}(f) && \wedge \\ & \mathtt{\_inj}(f) \end{aligned}$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$
\begin{aligned}
\forall a : \texttt{int} \cdot \exists b : \texttt{int}, f : \texttt{set}\,(\tau \times \texttt{int}) \cdot \\
& f \in S \leftrightarrow a..b && \wedge \\
& \forall x : \tau, y : \texttt{int}, z : \texttt{int} \cdot x \mapsto y \in f \wedge x \mapsto z \in f \Rightarrow y = z && \wedge \\
& S \subseteq \mathbf{dom}(f) && \wedge \\
& \_\texttt{inj}(f)
\end{aligned}
$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$
\begin{array}{ll}
\forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}\,(\tau \times \mathtt{int}) \cdot & \\
\qquad f \in S \leftrightarrow a..b & \wedge \\
\qquad \forall x : \tau, y : \mathtt{int}, z : \mathtt{int} \cdot x \mapsto y \in f \wedge x \mapsto z \in f \Rightarrow y = z & \wedge \\
\qquad S \subseteq \mathbf{dom}(f) & \wedge \\
\qquad \_\mathtt{inj}(f) &
\end{array}
$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$
\begin{aligned}
&\forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}\,(\tau \times \mathtt{int}) \cdot \\
&\qquad \forall x : \tau, y : \mathtt{int} \cdot x \mapsto y \in f \Rightarrow x \in S \land a \leq y \land y \leq b \qquad \land \\
&\qquad \forall x : \tau, y : \mathtt{int}, z : \mathtt{int} \cdot x \mapsto y \in f \land x \mapsto z \in f \Rightarrow y = z \qquad \land \\
&\qquad S \subseteq \mathbf{dom}(f) \qquad \land \\
&\qquad \_\mathtt{inj}(f)
\end{aligned}
$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$
\begin{aligned}
\forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}(\tau \times \mathtt{int}) \cdot \\
& \forall x : \tau, y : \mathtt{int} \cdot x \mapsto y \in f \Rightarrow x \in S \land a \leq y \land y \leq b && \land \\
& \forall x : \tau, y : \mathtt{int}, z : \mathtt{int} \cdot x \mapsto y \in f \land x \mapsto z \in f \Rightarrow y = z && \land \\
& S \subseteq \mathbf{dom}(f) && \land \\
& \_\mathtt{inj}(f)
\end{aligned}
$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$
\begin{aligned}
&\forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}(\tau \times \mathtt{int}) \cdot \\
&\qquad \forall x : \tau, y : \mathtt{int} \cdot x \mapsto y \in f \Rightarrow x \in S \land a \leq y \land y \leq b && \land \\
&\qquad \forall x : \tau, y : \mathtt{int}, z : \mathtt{int} \cdot x \mapsto y \in f \land x \mapsto z \in f \Rightarrow y = z && \land \\
&\qquad \forall z : \tau \cdot z \in S \Rightarrow \exists w : \mathtt{int} \cdot z \mapsto w \in f && \land \\
&\qquad \_\mathtt{inj}(f)
\end{aligned}
$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$
\begin{aligned}
\forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}(\tau \times \mathtt{int}) \cdot \\
& \forall x : \tau, y : \mathtt{int} \cdot x \mapsto y \in f \Rightarrow x \in S \land a \leq y \land y \leq b && \land \\
& \forall x : \tau, y : \mathtt{int}, z : \mathtt{int} \cdot x \mapsto y \in f \land x \mapsto z \in f \Rightarrow y = z && \land \\
& \forall z : \tau \cdot z \in S \Rightarrow \exists w : \mathtt{int} \cdot z \mapsto w \in f && \land \\
& \mathtt{\_inj}(f)
\end{aligned}
$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S is defined as follows in B:

$$
\begin{aligned}
&\forall a : \mathtt{int} \cdot \exists b : \mathtt{int}, f : \mathtt{set}\,(\tau \times \mathtt{int}) \cdot \\
&\qquad \forall x : \tau, y : \mathtt{int} \cdot x \mapsto y \in f \Rightarrow x \in S \land a \leq y \land y \leq b \qquad \land \\
&\qquad \forall x : \tau, y : \mathtt{int}, z : \mathtt{int} \cdot x \mapsto y \in f \land x \mapsto z \in f \Rightarrow y = z \qquad \land \\
&\qquad \forall z : \tau \cdot z \in S \Rightarrow \exists w : \mathtt{int} \cdot z \mapsto w \in f \qquad \land \\
&\qquad \forall x : \tau, y : \tau, z : \tau \cdot x \mapsto z \in f \land y \mapsto z \in f \Rightarrow x = y
\end{aligned}
$$

How large is the gain?

Let S be a set of elements of type τ.
The expression **finite** S can be encoded as follows:

$$\exists N \colon \mathtt{int}, f \colon \tau \to \mathtt{int} \cdot \\ \quad \forall x \colon \tau, y \colon \tau, z \colon \mathtt{int} \cdot f(x) = z \land f(y) = z \Rightarrow x = y \quad \land \\ \quad \forall x \colon \tau \cdot x \in S \Rightarrow 0 \leq f(x) \land f(x) < N$$

Does this work?

```
MACHINE
  M
VARIABLES
  s0
INVARIANT
  s0 ⊆ NAT ∧
  s0 ∩ (ℤ \ ℕ) ∈ FIN(ℤ)
INITIALISATION
  s0 :∈ 𝒫(NAT)
END
```

Does this work?

```
MACHINE
  M
VARIABLES
  s0
INVARIANT
  s0 ⊆ NAT ∧
  s0 ∩ (ℤ \ ℕ) ∈ FIN(ℤ)
INITIALISATION
  s0 :∈ 𝒫(NAT)
END
```

The following proof obligation is generated:

$$s0 \in \mathcal{P}(\mathsf{NAT}) \Rightarrow s0 \subseteq \mathsf{NAT} \land s0 \cap (\mathbb{Z} \setminus \mathbb{N}) \in \mathsf{FIN}(\mathbb{Z})$$

Does this work?

```
MACHINE
  M
VARIABLES
  s0
INVARIANT
  s0 ⊆ NAT ∧
  s0 ∩ (ℤ \ ℕ) ∈ FIN(ℤ)
INITIALISATION
  s0 :∈ 𝒫(NAT)
END
```

The following proof obligation is generated:

$$\mathtt{s0} \in \mathcal{P}(\mathbf{NAT}) \Rightarrow \mathtt{s0} \subseteq \mathbf{NAT} \wedge \mathtt{s0} \cap (\mathbb{Z} \setminus \mathbb{N}) \in \mathbf{FIN}(\mathbb{Z})$$

which boils down to proving:

$$\mathtt{s0} \in \mathcal{P}(\mathbf{NAT}) \Rightarrow \mathtt{s0} \in \mathbf{FIN}(\mathbb{Z})$$

Does this work?

```
MACHINE
  M
VARIABLES
  s0
INVARIANT
  s0 ⊆ NAT ∧
  s0 ∩ (ℤ \ ℕ) ∈ FIN(ℤ)
INITIALISATION
  s0 :∈ 𝒫(NAT)
END
```

The following proof obligation is generated:

$$s0 \in \mathcal{P}(\mathsf{NAT}) \Rightarrow s0 \subseteq \mathsf{NAT} \wedge s0 \cap (\mathbb{Z} \setminus \mathbb{N}) \in \mathsf{FIN}(\mathbb{Z})$$

which boils down to proving:

$$s0 \in \mathcal{P}(\mathsf{NAT}) \Rightarrow s0 \in \mathsf{FIN}(\mathbb{Z})$$

✗ predicate prover from Atelier B

Does this work?

```
MACHINE
  M
VARIABLES
  s0
INVARIANT
  s0 ⊆ NAT ∧
  s0 ∩ (ℤ \ ℕ) ∈ FIN(ℤ)
INITIALISATION
  s0 :∈ 𝒫(NAT)
END
```

The following proof obligation is generated:

$$\mathtt{s0} \in \mathcal{P}(\mathbf{NAT}) \Rightarrow \mathtt{s0} \subseteq \mathbf{NAT} \land \mathtt{s0} \cap (\mathbb{Z} \setminus \mathbb{N}) \in \mathbf{FIN}(\mathbb{Z})$$

which boils down to proving:

$$\mathtt{s0} \in \mathcal{P}(\mathbf{NAT}) \Rightarrow \mathtt{s0} \in \mathbf{FIN}(\mathbb{Z})$$

- ✗ predicate prover from Atelier B
- ✗ CVC5 with ppTrans

Does this work?

```
MACHINE
  M
VARIABLES
  s0
INVARIANT
  s0 ⊆ NAT ∧
  s0 ∩ (ℤ \ ℕ) ∈ FIN(ℤ)
INITIALISATION
  s0 :∈ 𝒫(NAT)
END
```

The following proof obligation is generated:

$$\texttt{s0} \in \mathcal{P}(\mathbf{NAT}) \Rightarrow \texttt{s0} \subseteq \mathbf{NAT} \land \texttt{s0} \cap (\mathbb{Z} \setminus \mathbb{N}) \in \mathbf{FIN}(\mathbb{Z})$$

which boils down to proving:

$$\texttt{s0} \in \mathcal{P}(\mathbf{NAT}) \Rightarrow \texttt{s0} \in \mathbf{FIN}(\mathbb{Z})$$

- ✗ predicate prover from Atelier B
- ✗ CVC5 with ppTrans
- ✓ CVC5 with

BEer: encoding B POs in SMT-LIB using HO

In the current state of :

ppTrans	unsat	sat	unknown	Total
unsat	14,831	0	**1,062**	15,893
sat	0	0	0	0
unknown	**272**	0	780	1,052
Total	15,103	0	1,842	**16,945**

Benchmark specs:

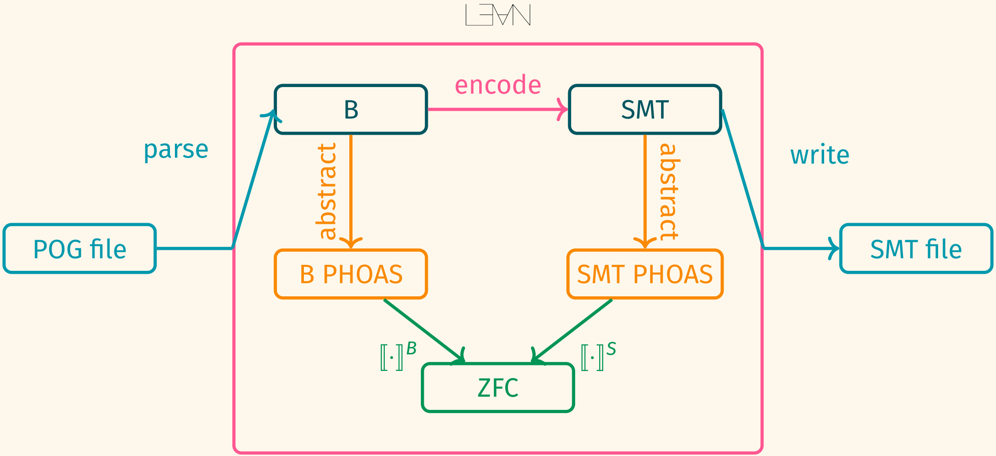
- **681,285** POs in total
- Apple M2 (10 CPU cores, 24 GB RAM)
- CVC5 with incremental mode, MBQI enabled and 3s timeout per query

BEer: encoding B POs in SMT-LIB using HO

Demo

Demo time!

Architecture of 🍺



BEer: encoding B POs in SMT-LIB using HO

We define a denotation function for abstract B terms:

$$[\![\cdot]\!]^{\mathsf{B}} \quad : \quad \mathtt{Term}\,\mathcal{V} \to \mathcal{V}$$

BEer: encoding B POs in SMT-LIB using HO

We define a denotation function for abstract B terms:

$$[\![\cdot]\!]^{\mathsf{B}} \quad : \quad \mathtt{Term}\,\mathtt{ZFSet} \rightarrow \mathtt{ZFSet}$$

BEer: encoding B POs in SMT-LIB using HO

We define a denotation function for abstract B terms:

$$[\![\cdot]\!]^{B} \quad : \quad \mathtt{Term}\,\mathtt{Dom} \rightarrow \mathtt{Dom}$$

BEer: encoding B POs in SMT-LIB using HO

We define a denotation function for abstract B terms:

$$[\![\cdot]\!]^{\mathsf{B}} \quad : \quad \mathtt{Term}\,\mathtt{Dom} \to \mathtt{Dom}$$

where

$$\mathtt{Dom} := \sum_{x,\tau} x \in [\![\tau]\!]^{\mathsf{Z}}$$

$$\begin{cases} [\![\mathit{int}]\!]^{\mathsf{Z}} & := \mathbb{Z}^{\mathsf{Z}} \\ [\![\mathit{bool}]\!]^{\mathsf{Z}} & := \mathbb{B}^{\mathsf{Z}} \\ [\![\mathit{set}\ \alpha]\!]^{\mathsf{Z}} & := \mathcal{P}^{\mathsf{Z}}([\![\alpha]\!]^{\mathsf{Z}}) \\ [\![\alpha \times^{\mathsf{B}} \beta]\!]^{\mathsf{Z}} & := [\![\alpha]\!]^{\mathsf{Z}} \times^{\mathsf{Z}} [\![\beta]\!]^{\mathsf{Z}} \end{cases}$$

What we are after

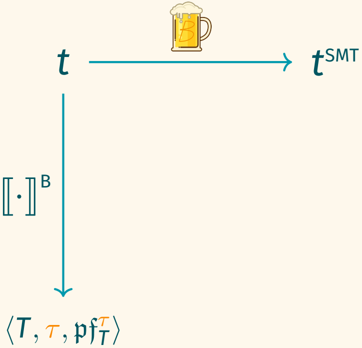
What we are after

t

What we are after

$$t \longrightarrow t^{\mathsf{SMT}}$$

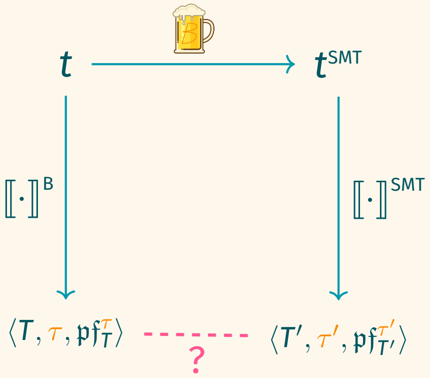
What we are after



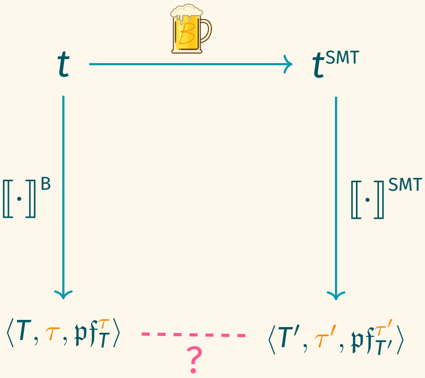
What we are after

$$\begin{array}{ccc} t & \xrightarrow{\qquad\qquad} & t^{\mathsf{SMT}} \\ \Big\downarrow{\scriptstyle [\![\cdot]\!]^{\mathsf{B}}} & & \Big\downarrow{\scriptstyle [\![\cdot]\!]^{\mathsf{SMT}}} \\ \langle T, \tau, \mathfrak{pf}_T^{\tau} \rangle & & \langle T', \tau', \mathfrak{pf}_{T'}^{\tau'} \rangle \end{array}$$

What we are after



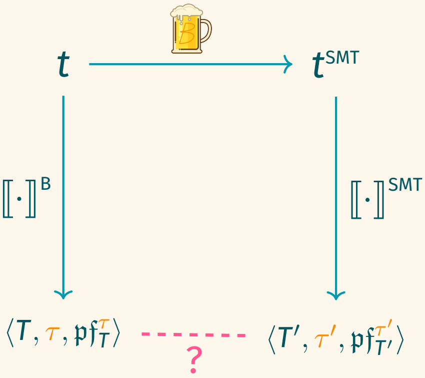
What we are after



We should at least have:

$$\tau' = \tau^{\mathsf{SMT}}$$

What we are after



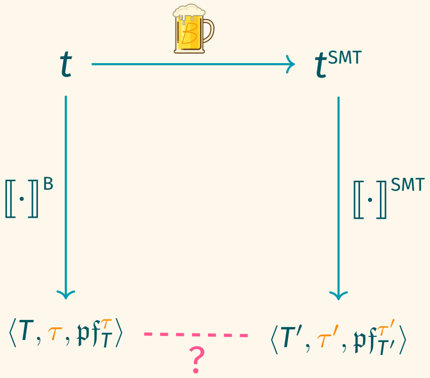
We should at least have:

$$\tau' = \tau^{\mathsf{SMT}}$$

and something like:

T "corresponds to" T'

What we are after



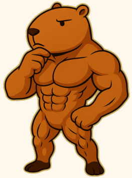
We should at least have:

$$\tau' = \tau^{\mathsf{SMT}}$$

and something like:

T "corresponds to" T'

which has to be formalized.



Theorem

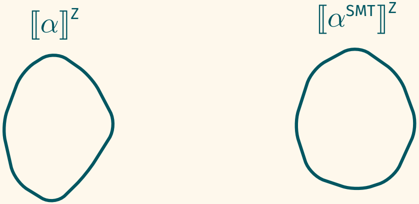
B types and their SMT-LIB translations are isomorphic, i.e., for any B type α:

$$[\![\alpha]\!]^{\mathsf{Z}} \cong [\![\alpha^{\mathsf{SMT}}]\!]^{\mathsf{Z}}$$

Theorem

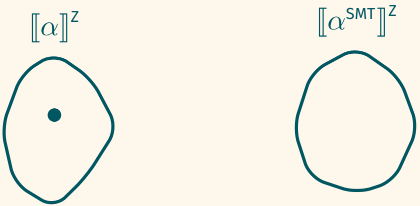
B types and their SMT-LIB translations are isomorphic, i.e., for any B type α:

$$[\![\alpha]\!]^{\mathsf{Z}} \cong [\![\alpha^{\mathsf{SMT}}]\!]^{\mathsf{Z}}$$



Theorem

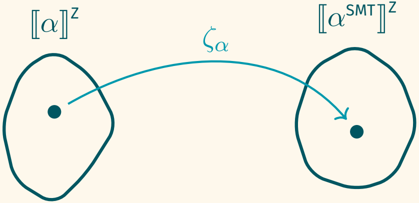
> *B types and their SMT-LIB translations are isomorphic, i.e., for any B type α:*
>
> $$[\![\alpha]\!]^{\mathsf{Z}} \cong [\![\alpha^{\mathsf{SMT}}]\!]^{\mathsf{Z}}$$



Theorem

B types and their SMT-LIB translations are isomorphic, i.e., for any B type α:

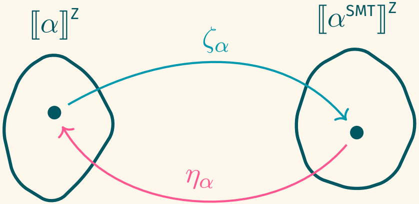
$$[\![\alpha]\!]^{\mathsf{Z}} \cong [\![\alpha^{\mathsf{SMT}}]\!]^{\mathsf{Z}}$$



Theorem

B types and their SMT-LIB translations are isomorphic, i.e., for any B type α:

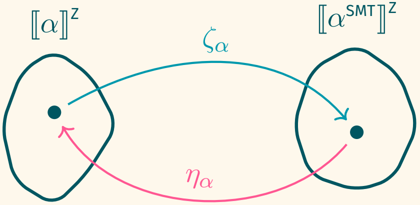
$$[\![\alpha]\!]^{Z} \cong [\![\alpha^{SMT}]\!]^{Z}$$



Theorem

B types and their SMT-LIB translations are isomorphic, i.e., for any B type α:

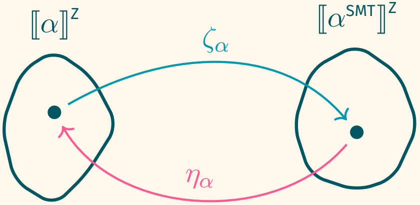
$$[\![\alpha]\!]^{\mathsf{Z}} \cong [\![\alpha^{\mathsf{SMT}}]\!]^{\mathsf{Z}}$$



Inductively-defined indexed family of canonical isomorphisms $(\zeta_\alpha)_{\alpha\,:\,\mathtt{BType}}$ with associated retractions $(\eta_\alpha)_{\alpha\,:\,\mathtt{BType}}$.

Theorem

> *B types and their SMT-LIB translations are isomorphic, i.e., for any B type α:*
>
> $$[\![\alpha]\!]^{\mathsf{Z}} \cong [\![\alpha^{\mathsf{SMT}}]\!]^{\mathsf{Z}}$$



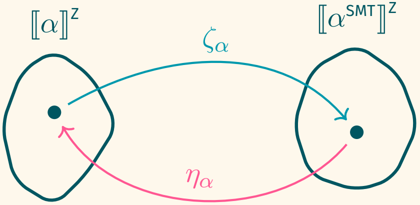
Inductively-defined indexed family of canonical isomorphisms $(\zeta_\alpha)_{\alpha\,:\,\mathtt{BType}}$ with associated retractions $(\eta_\alpha)_{\alpha\,:\,\mathtt{BType}}$.

$$\eta_\alpha \circ \zeta_\alpha = \mathbb{1}_{[\![\alpha]\!]^{\mathsf{Z}}}$$

Theorem

B types and their SMT-LIB translations are isomorphic, i.e., for any B type α:

$$[\![\alpha]\!]^{\mathsf{Z}} \cong [\![\alpha^{\mathsf{SMT}}]\!]^{\mathsf{Z}}$$



Inductively-defined indexed family of canonical isomorphisms $(\zeta_\alpha)_{\alpha\,:\,\mathsf{BType}}$ with associated retractions $(\eta_\alpha)_{\alpha\,:\,\mathsf{BType}}$.

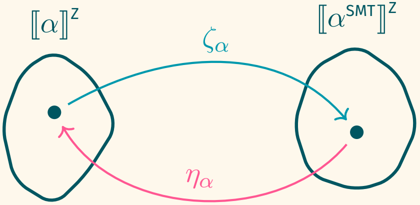
$$\eta_\alpha \circ \zeta_\alpha = \mathbb{1}_{[\![\alpha]\!]^{\mathsf{Z}}}$$

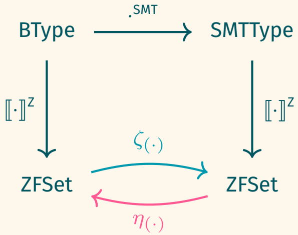
Yes, η_α is ζ_α^{-1} but we define it constructively as well.

Theorem

B types and their SMT-LIB translations are isomorphic, i.e., for any B type α:

$$[\![\alpha]\!]^{\mathsf{Z}} \cong [\![\alpha^{\mathsf{SMT}}]\!]^{\mathsf{Z}}$$



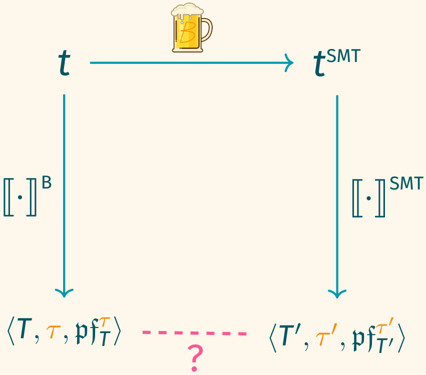


Inductively-defined indexed family of canonical isomorphisms $(\zeta_\alpha)_{\alpha\,:\,\mathsf{BType}}$ with associated retractions $(\eta_\alpha)_{\alpha\,:\,\mathsf{BType}}$.

$$\eta_\alpha \circ \zeta_\alpha = \mathbb{1}_{[\![\alpha]\!]^{\mathsf{Z}}}$$

Yes, η_α is ζ_α^{-1} but we define it constructively as well.

~~What we are after~~ Found it!



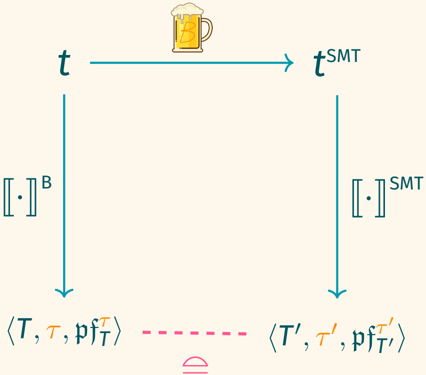
~~What we are after~~ Found it!

$$t \longrightarrow t^{\mathsf{SMT}}$$

$$[\![\cdot]\!]^{\mathsf{B}} \downarrow \qquad \downarrow [\![\cdot]\!]^{\mathsf{SMT}}$$

$$\langle T, \tau, \mathfrak{pf}_T^{\tau} \rangle \; \text{- - - - - -} \; \langle T', \tau', \mathfrak{pf}_{T'}^{\tau'} \rangle$$

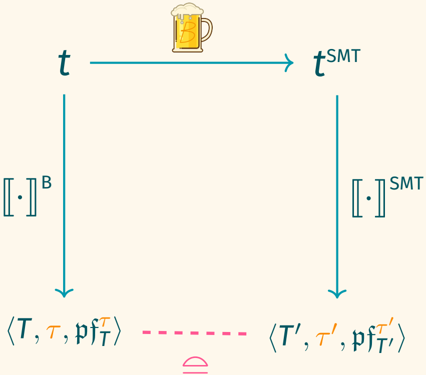
~~What we are after~~ Found it!



We define $\langle T, \tau, \mathfrak{pf}_T^{\tau} \rangle \mathrel{\stackrel{\frown}{=}} \langle T', \tau', \mathfrak{pf}_{T'}^{\tau'} \rangle$ as:

$$\tau' = \tau^{\mathsf{SMT}} \land \eta_{\tau}(T') = T$$

~~What we are after~~ Found it!



We define $\langle T, \tau, \mathfrak{pf}_T^{\tau} \rangle \mathrel{\widehat{=}} \langle T', \tau', \mathfrak{pf}_{T'}^{\tau'} \rangle$ as:

$$\tau' = \tau^{\mathsf{SMT}} \wedge \eta_\tau(T') = T$$

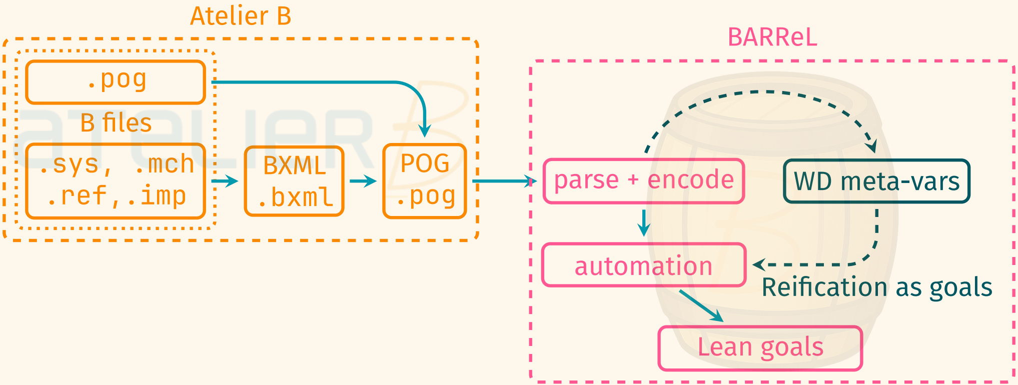
Now prove that this is preserved!

BARReL: when the glass is empty

B Automated tRanslation for Reasoning in Lean

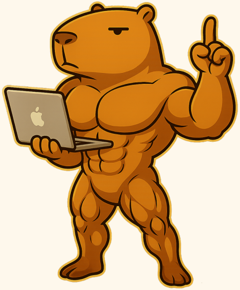
BARReL: when the glass is empty

Pipeline



BARReL: when the glass is empty

Demo



Demo time!

Conclusion

Contributions:

- **Higher-order encoding** leveraging SMT-LIB's latest features
- **Formal semantics** for subsets of B proof obligations and SMT-LIB
- **Loosening** B types to reconcile set-theoretic and type-theoretic notions
- **ZFLean**: framework for set-level developments in Lean

Current/future work:

- **Correctness** of the encoding

`VTrelat/{BEer, BARReL, ZFLean}`