Safely Encoding B Proof Obligations in SMT-LIB

Final EuroProofNet Symposium
WG2: Workshop on Automated Reasoning and Proof Logging

Vincent Trélat

Université de Lorraine, CNRS, INRIA, Loria, Nancy, France

September 12, 2025

Introduction

B method

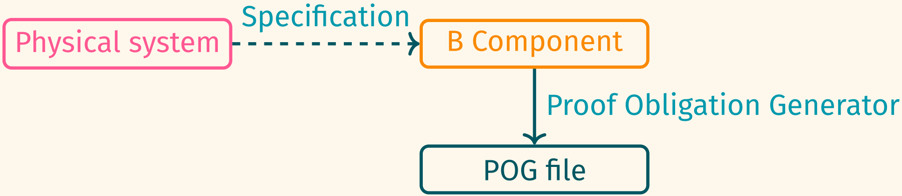
Physical system

Introduction

B method

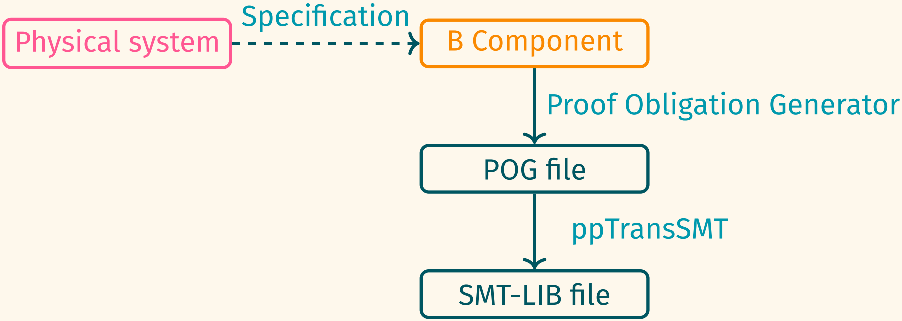
Introduction

B method



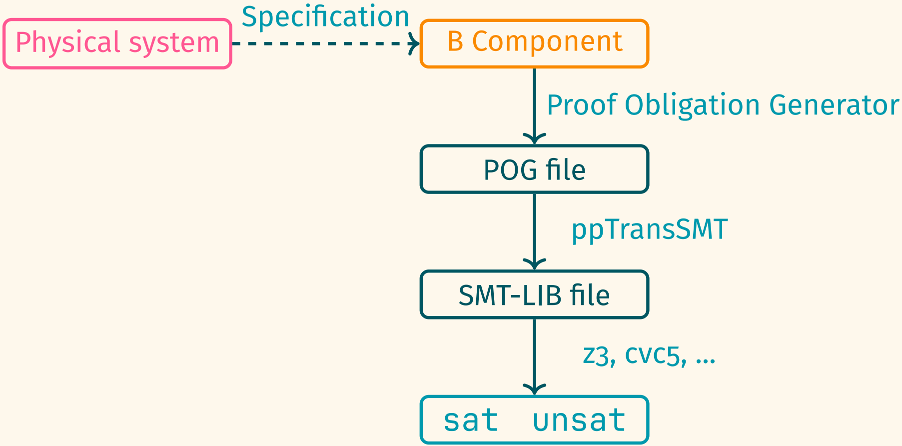
Introduction

B method



Introduction

B method



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)

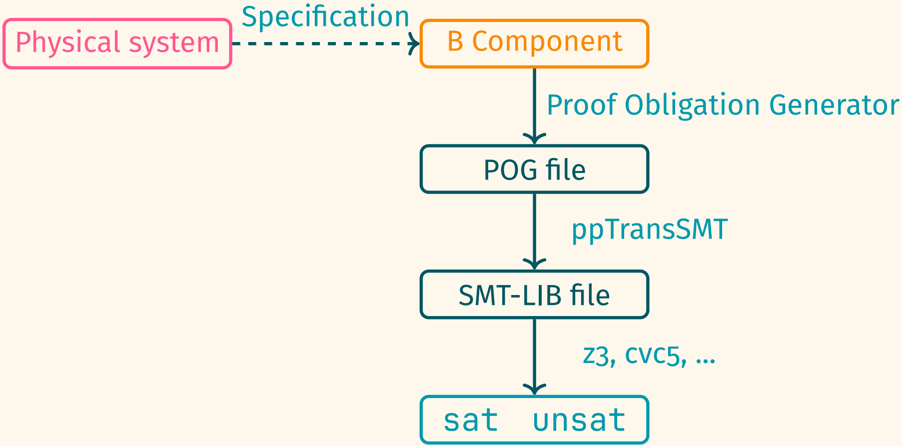
- Standard input format for SMT solvers (e.g. z3, cvc5, veriT)
- Based on many-sorted **first-order logic**
- Comes with many theories (e.g. arrays, integer and real arithmetic)

SMT-LIB v2.7

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

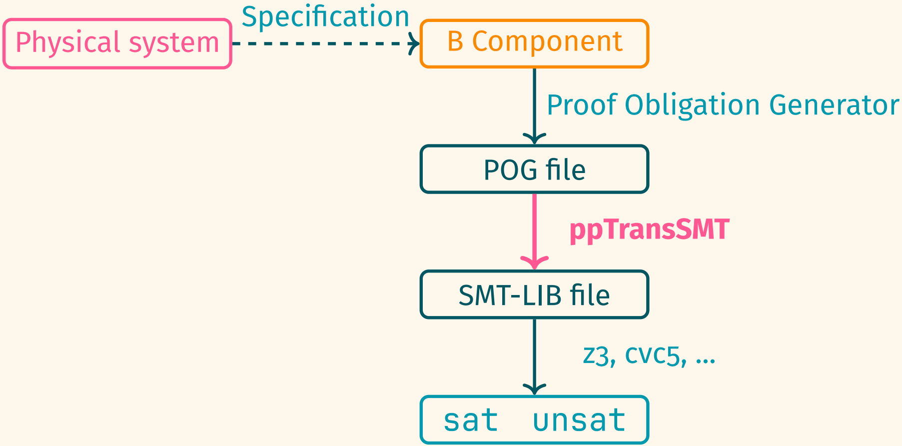
Introduction

B method



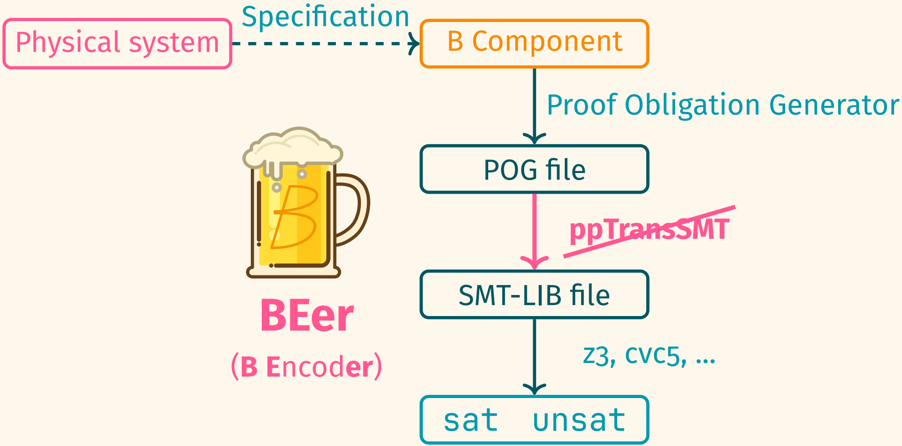
Introduction

B method



Introduction

B method



Introduction

Because Germany taught me to lean on beer,

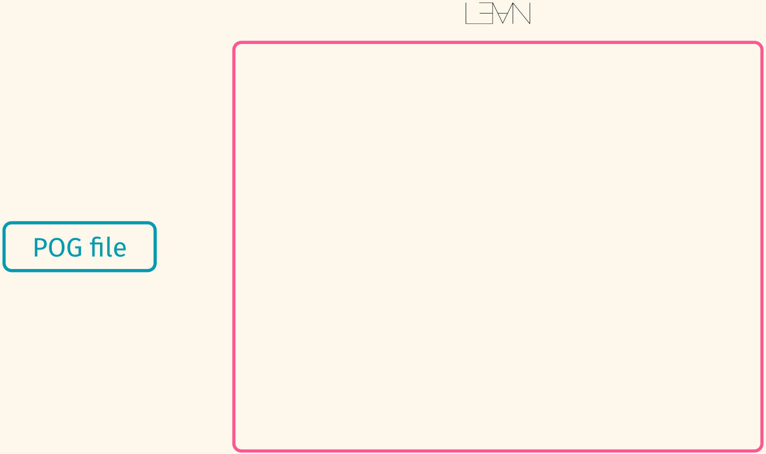
Introduction

Because Germany taught me to lean on beer,

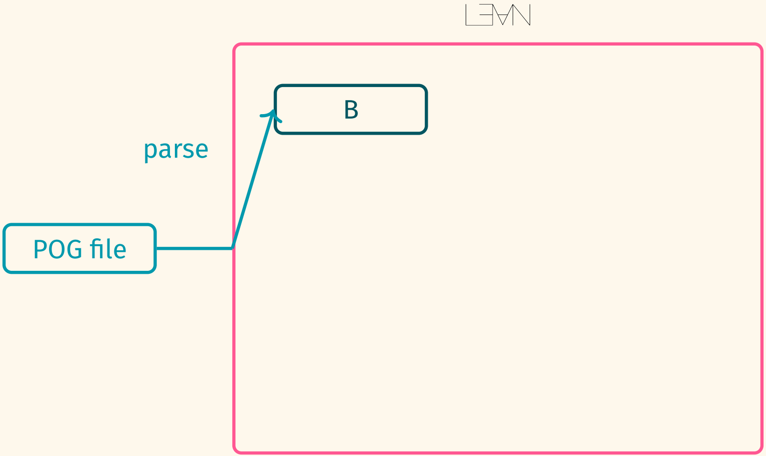
is written in LEAN

Architecture of

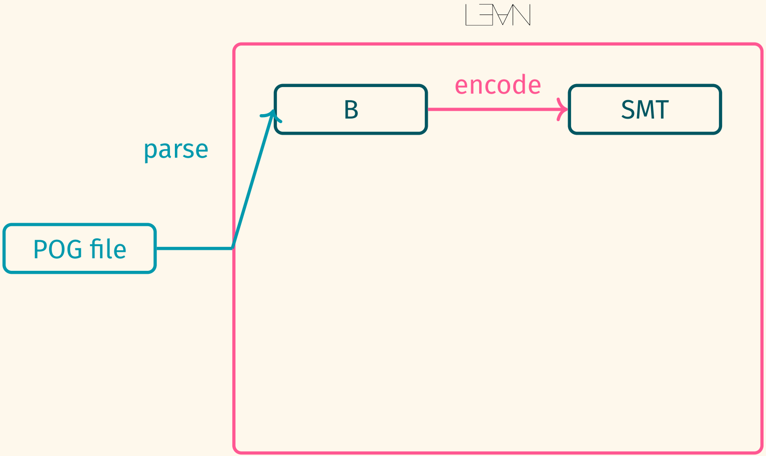
Architecture of



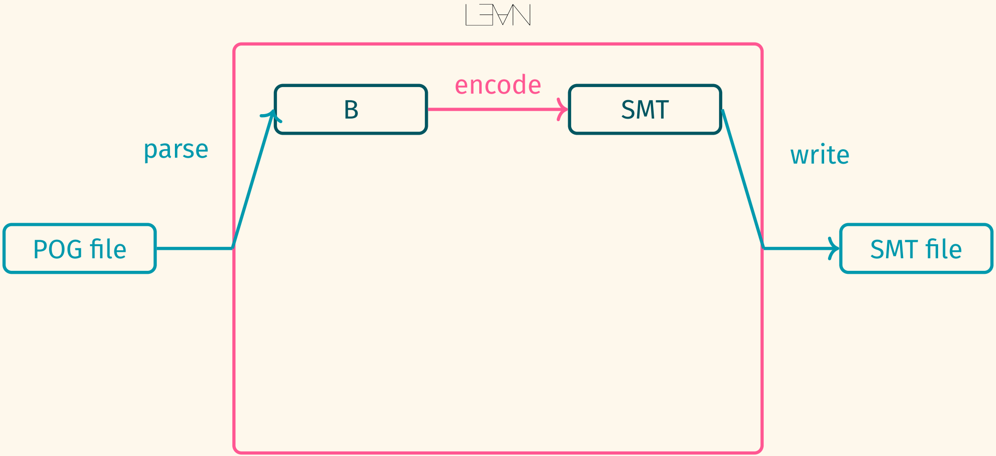
Architecture of 🍺



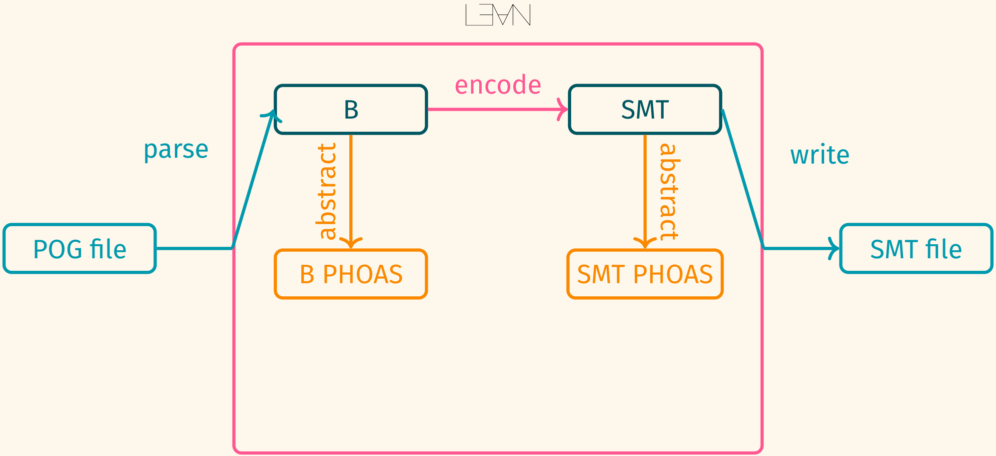
Architecture of 🍺



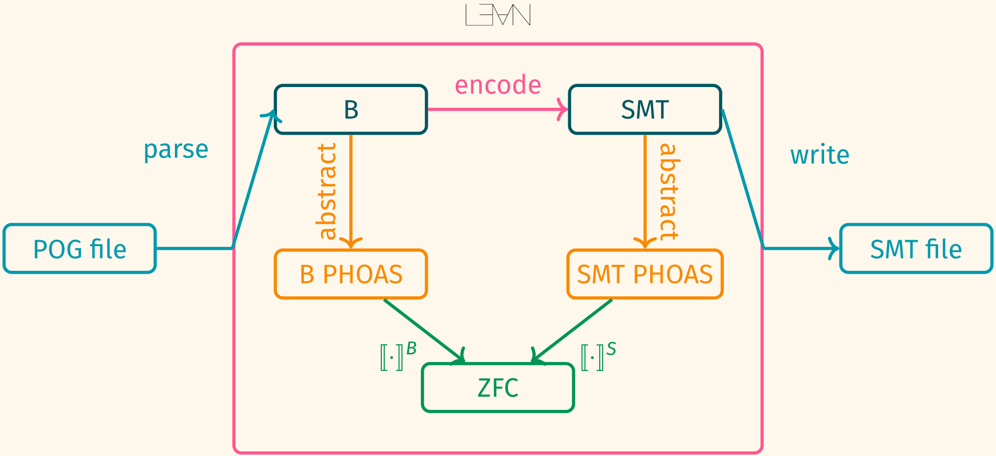
Architecture of 🍺



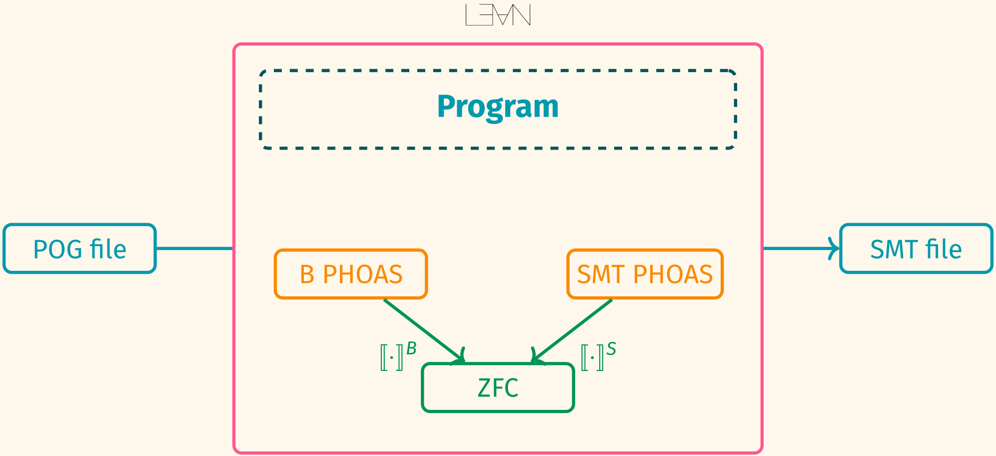
Architecture of 🍺



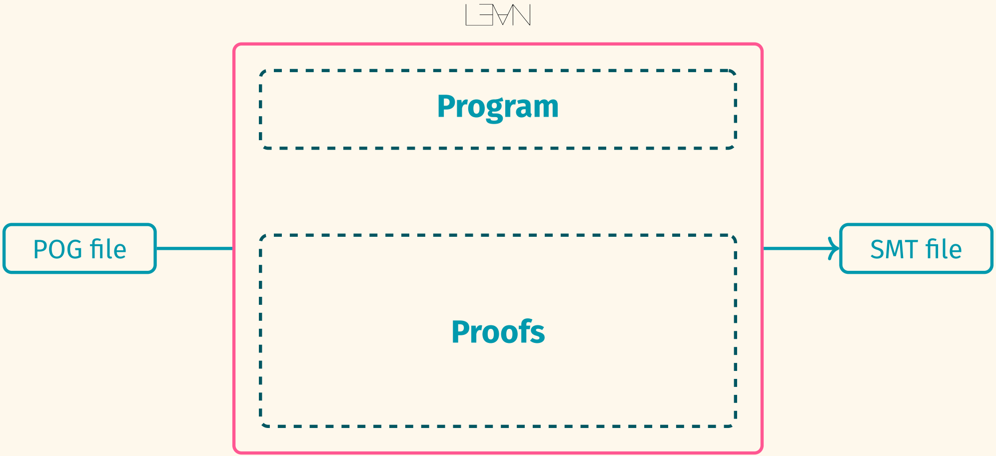
Architecture of 🍺



Architecture of 🍺



Architecture of 🍺



Overview of the encoding

(former) first-order encoding

(new) higher-order encoding 🍺

Overview of the encoding

(former) first-order encoding

- FOL

(new) higher-order encoding 🍺

- HOL

Overview of the encoding

(former) first-order encoding

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

(new) higher-order encoding 🍺

- HOL
- Definition of sets via characteristic predicates

Overview of the encoding

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

Overview of the encoding

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

(former) first-order 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 of the encoding

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

(former) first-order 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)))))
```

(new) higher-order 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 of the encoding

(former) first-order encoding

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

(new) higher-order encoding 🍺

- HOL
- Definition of sets via characteristic predicates

Overview of the encoding

(former) first-order encoding

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

(new) higher-order encoding

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

Overview of the encoding

(former) first-order encoding

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

(new) higher-order 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 $f \in A \nrightarrow B$.

(former) first-order encoding

f is a relation between A and B:

$$f \subseteq A \times B$$

f is functional:

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

Suppose we have a function $f \in A \mathrel{+\!\!\!\!\!\rightarrow} B$.

(former) first-order encoding

f is a relation between A and B:

$$f \subseteq A \times B$$

f is functional:

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

(new) higher-order encoding

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

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

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

(former) first-order encoding

f is a relation between A and B:

$$f \subseteq A \times B$$

f is functional:

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

(new) higher-order encoding 🍺

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

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

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

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

(former) first-order 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 $f \in A \nrightarrow B$. Let τ_A and τ_B represent the types of A and B respectively.

(former) first-order encoding

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

(new) higher-order encoding

```
(declare-const f (→ τA (Option τB)))
```

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

(former) first-order encoding

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

(new) higher-order encoding

```
(declare-const f (→ τA (Option τB)))
```

+ 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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

How large is the gain?

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

$$
\begin{aligned}
&\exists N : \mathtt{int}, f : \tau \to \mathtt{int} \cdot \\
&\qquad \forall x : \tau, y : \tau, z : \mathtt{int} \cdot f(x) = z \land f(y) = z \Rightarrow x = y \qquad \land \\
&\qquad \forall x : \tau \cdot x \in S \Rightarrow 0 \leq f(x) \land f(x) < N
\end{aligned}
$$

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:

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

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:

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

which boils down to proving:

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

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:

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

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

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 ppTransSMT

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 ppTransSMT

✓ CVC5 with 

Results

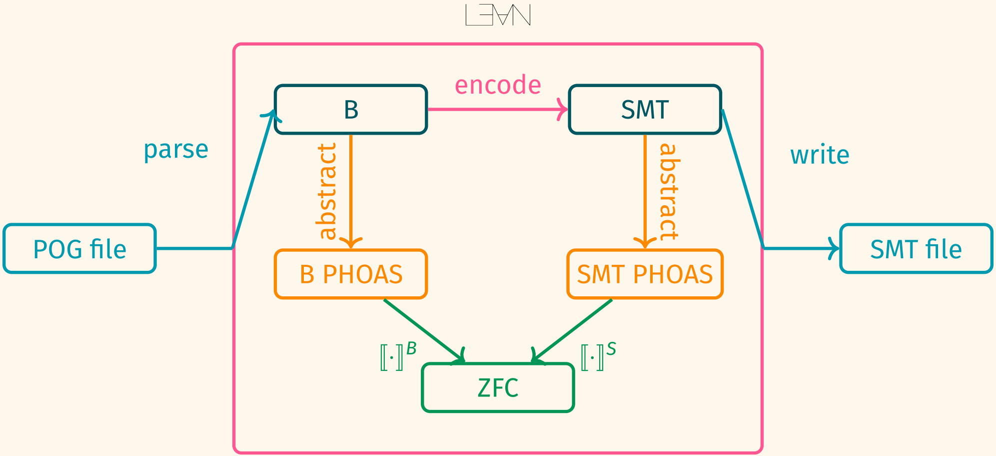
In the current state of we have:

ppTrans \	unsat	sat	unknown	Total
unsat	14,457	0	**431**	14,888
sat	1	1	5	7
unknown	**236**	5	472	713
Total	14,694	6	908	**15,608**

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

Architecture of 🍺



Roadmap

- ☐ Syntax
- ☐ Type system
- ☐ Semantics

Concrete terms

Concrete terms are **low-level** syntactic constructs used in the implementation of the encoding.

Concrete terms

Concrete terms are **low-level** syntactic constructs used in the implementation of the encoding.

```
inductive Term where
  | var (v : String) | int (n : Int) | bool (b : Bool) | x ↦ᴮ y
  -- arithmetic
  | x +ᴮ y | x -ᴮ y | x *ᴮ y | x ≤ᴮ y
  -- logic
  | x ∧ᴮ y | ¬ᴮ x | x =ᴮ y | ∀ᴮ (vs : List String) ∈ᴮ D ·P
  -- set operations
  | ℤ | 𝔹 | x ∈ᴮ y | 𝒫ᴮ S | S ×ᴮ T | S ∪ᴮ T | S ∩ᴮ T | |S|ᴮ
  | {(vs : List String) ∈ᴮ D | P}
  -- functions
  | app (f x : Term) | A ⇸ᴮ B | min (S : Term) | max (S : Term)
  | λᴮ (vs : List String) ∈ᴮ D | f
```

Concrete terms

Example

$$\lambda^{\mathsf{B}}[] \in^{\mathsf{B}} \mathit{int}\,2 \mid \mathit{var}\,\text{"}x\text{"} =^{\mathsf{B}} (\mathit{int}\,0 \wedge^{\mathsf{B}} \mathbb{B}) \qquad \mathit{int}\,0 +^{\mathsf{B}} \mathit{int}\,1 \in^{\mathsf{B}} \mathbb{Z}$$

$$\forall^{\mathsf{B}}[\text{"}x\text{"}, \text{"}y\text{"}] \in^{\mathsf{B}} \mathbb{Z} \times^{\mathsf{B}} \mathbb{Z} \cdot \mathit{var}\,\text{"}x\text{"} \leq^{\mathsf{B}} \mathit{var}\,\text{"}y\text{"} \qquad \mid \mathbb{Z} \mid^{\mathsf{B}} \in^{\mathsf{B}} \mathbb{Z}$$

are all syntactically valid terms.[1]

[1] Read with usual priorities.

Concrete terms

Example

$$\lambda^{B}[] \in^{B} \mathit{int}\,2 \mid \mathit{var}\,"x" =^{B} (\mathit{int}\,0 \wedge^{B} \mathbb{B}) \qquad \mathit{int}\,0 +^{B} \mathit{int}\,1 \in^{B} \mathbb{Z}$$

$$\forall^{B}["x", "y"] \in^{B} \mathbb{Z} \times^{B} \mathbb{Z} \cdot \mathit{var}\,"x" \leq^{B} \mathit{var}\,"y" \qquad |\,\mathbb{Z}\,|^{B} \in^{B} \mathbb{Z}$$

are all syntactically valid terms.[1]

Some of these terms are type-correct,

[1] Read with usual priorities.

Concrete terms

Example

$$\lambda^{B}[] \in^{B} \mathit{int}\,2 \mid \mathit{var}\,\text{"x"} =^{B} (\mathit{int}\,0 \wedge^{B} \mathbb{B}) \qquad \mathit{int}\,0 +^{B} \mathit{int}\,1 \in^{B} \mathbb{Z}$$

$$\forall^{B}[\text{"x"}, \text{"y"}] \in^{B} \mathbb{Z} \times^{B} \mathbb{Z} \cdot \mathit{var}\,\text{"x"} \leq^{B} \mathit{var}\,\text{"y"} \qquad \mid \mathbb{Z} \mid^{B} \in^{B} \mathbb{Z}$$

are all syntactically valid terms.[1]

Some of these terms are type-correct, well-formed,

[1] Read with usual priorities.

Concrete terms

Example

$$\lambda^{B}[] \in^{B} \mathit{int}\,2 \mid \mathit{var}\,\text{"x"} =^{B} (\mathit{int}\,0 \wedge^{B} \mathbb{B}) \qquad \mathit{int}\,0 +^{B} \mathit{int}\,1 \in^{B} \mathbb{Z}$$

$$\forall^{B}[\text{"x"}, \text{"y"}] \in^{B} \mathbb{Z} \times^{B} \mathbb{Z} \cdot \mathit{var}\,\text{"x"} \leq^{B} \mathit{var}\,\text{"y"} \qquad |\,\mathbb{Z}\,|^{B} \in^{B} \mathbb{Z}$$

are all syntactically valid terms.[1]

Some of these terms are type-correct, well-formed, or do not make sense.

[1]Read with usual priorities.

Abstract terms

We define **abstract terms** as a layer of abstraction over concrete terms.

Abstract terms

We define **abstract terms** as a layer of abstraction over concrete terms.

- ease formal reasoning about the language via higher-level constructs
- bridge the gap between a deeply embedded language and Lean

Abstract terms

We define **abstract terms** as a layer of abstraction over concrete terms.

- ease formal reasoning about the language via higher-level constructs
- bridge the gap between a deeply embedded language and Lean

Example

> In $\forall^{B}["x"] \in^{B} D \cdot P$, P is only a term, not a predicate.

Abstract terms

We define **abstract terms** as a layer of abstraction over concrete terms.

- ease formal reasoning about the language via higher-level constructs
- bridge the gap between a deeply embedded language and Lean

Example

In $\forall^{\mathsf{B}}[\text{"}x\text{"}] \in^{\mathsf{B}} D \cdot P$, P is only a term, not a predicate.

Remark

α-renaming and substitutions have to be handled explicitly:

$$(\lambda^{\mathsf{B}}[\text{"}x\text{"}] \in^{\mathsf{B}} \mathbb{Z} \mid var\ \text{"}x\text{"}) \neq (\lambda^{\mathsf{B}}[\text{"}y\text{"}] \in^{\mathsf{B}} \mathbb{Z} \mid var\ \text{"}y\text{"})$$

$$(\lambda^{\mathsf{B}}[\text{"}x\text{"}] \in^{\mathsf{B}} \mathbb{Z} \mid var\ \text{"}x\text{"} +^{\mathsf{B}} var\ \text{"}y\text{"})\ [\text{"}y\text{"} := \text{"}x\text{"}] = ???$$

Abstract terms

PHOAS

Variable management is delegated to the underlying formal system (Lean):

```
bbinder : List String → Term → Term → Term
```

Abstract terms

PHOAS

Variable management is delegated to the underlying formal system (Lean):

```lean
bbinder : Term → (List String → Term) → Term
```

Abstract terms

PHOAS

Variable management is delegated to the underlying formal system (Lean):

```
LEAN
bbinder : Term 𝒱 → (List 𝒱 → Term 𝒱) → Term 𝒱
```

Abstract terms

PHOAS

Variable management is delegated to the underlying formal system (Lean):

```lean
bbinder {n} : Term 𝒱 → ((Fin n → 𝒱) → Term 𝒱) → Term 𝒱
```

Abstract terms

PHOAS

Variable management is delegated to the underlying formal system (Lean):

```
bbinder {n} : Term 𝒱 → ((Fin n → 𝒱) → Term 𝒱) → Term 𝒱
```

Example

Concrete term:

```
∀ᴮ ["x", "y"] ∈ᴮ ℤ ×ᴮ ℤ ·
  var "x" =ᴮ var "y"
```

Abstract term:

```
∀ᴮ ℤ ×ᴮ ℤ · fun x y ↦
  var x =ᴮ var y
```

Abstract terms

Abstraction function

Abstraction function: maps concrete terms to abstract terms under a renaming context

- Intuition: almost an identity function

Abstract terms

Abstraction function

Abstraction function: maps concrete terms to abstract terms under a renaming context

- Intuition: almost an identity function
- Implementation: a little more complex

Abstract terms

Abstraction function

Abstraction function: maps concrete terms to abstract terms under a renaming context

- Intuition: almost an identity function
- Implementation: a little more complex

Notation

> For any concrete term t and renaming context $\Delta : \mathtt{String} \to \mathtt{Option}\ \mathcal{V}$, the abstraction of t under Δ is denoted by $(\!| \ t \ |\!)_{\Delta}$.

Abstract terms

Abstraction function

Example

Let $Y \in \mathcal{V}$ and $\Delta := \{\text{"}y\text{"} \mapsto Y\}$. Consider the concrete term $\forall^{\mathsf{B}}[\text{"}x\text{"}] \in^{\mathsf{B}} \mathbb{Z} \cdot \mathit{var}\ \text{"}x\text{"} =^{\mathsf{B}} \mathit{var}\ \text{"}y\text{"}$.

$$\llparenthesis\ \forall^{\mathsf{B}}[\text{"}x\text{"}] \in^{\mathsf{B}} \mathbb{Z} \cdot \mathit{var}\ \text{"}x\text{"} =^{\mathsf{B}} \mathit{var}\ \text{"}y\text{"}\ \rrparenthesis_{\Delta}$$

Abstract terms

Abstraction function

Example

Let $Y \in \mathcal{V}$ and $\Delta := \{"y" \mapsto Y\}$. Consider the concrete term $\forall^{B}["x"] \in^{B} \mathbb{Z} \cdot \mathit{var}\ "x" =^{B} \mathit{var}\ "y"$.

$$\begin{aligned} &(\!| \forall^{B}["x"] \in^{B} \mathbb{Z} \cdot \mathit{var}\ "x" =^{B} \mathit{var}\ "y" |\!)_{\Delta} \\ &= \forall^{B} (\!| \mathbb{Z} |\!)_{\Delta} \cdot \left(X \mapsto (\!| \mathit{var}\ "x" =^{B} \mathit{var}\ "y" |\!)_{\Delta["x" \mapsto X]} \right) \end{aligned}$$

Abstract terms

Abstraction function

Example

Let $Y \in \mathcal{V}$ and $\Delta := \{\text{"}y\text{"} \mapsto Y\}$. Consider the concrete term $\forall^{\mathsf{B}}[\text{"}x\text{"}] \in^{\mathsf{B}} \mathbb{Z} \cdot \mathit{var}\ \text{"}x\text{"} =^{\mathsf{B}} \mathit{var}\ \text{"}y\text{"}$.

$$\begin{aligned}
&\llparenthesis\ \forall^{\mathsf{B}}[\text{"}x\text{"}] \in^{\mathsf{B}} \mathbb{Z} \cdot \mathit{var}\ \text{"}x\text{"} =^{\mathsf{B}} \mathit{var}\ \text{"}y\text{"}\ \rrparenthesis_{\Delta} \\
&= \forall^{\mathsf{B}} \llparenthesis\ \mathbb{Z}\ \rrparenthesis_{\Delta} \cdot \left(X \mapsto \llparenthesis\ \mathit{var}\ \text{"}x\text{"} =^{\mathsf{B}} \mathit{var}\ \text{"}y\text{"}\ \rrparenthesis_{\Delta[\text{"}x\text{"} \mapsto X]}\right) \\
&= \forall^{\mathsf{B}} \mathbb{Z} \cdot (X \mapsto \mathit{var}\, X =^{\mathsf{B}} \mathit{var}\, Y)
\end{aligned}$$

Abstract terms

Abstraction function

Example

Let $Y \in \mathcal{V}$ and $\Delta := \{\text{"y"} \mapsto Y\}$. Consider the concrete term $\forall^{\mathsf{B}}[\text{"x"}] \in^{\mathsf{B}} \mathbb{Z} \cdot \mathit{var}\ \text{"x"} =^{\mathsf{B}} \mathit{var}\ \text{"y"}$.

$$
\begin{aligned}
&\llparenthesis\, \forall^{\mathsf{B}}[\text{"x"}] \in^{\mathsf{B}} \mathbb{Z} \cdot \mathit{var}\ \text{"x"} =^{\mathsf{B}} \mathit{var}\ \text{"y"} \,\rrparenthesis_{\Delta} \\
&= \forall^{\mathsf{B}} \llparenthesis\, \mathbb{Z} \,\rrparenthesis_{\Delta} \cdot \left(X \mapsto \llparenthesis\, \mathit{var}\ \text{"x"} =^{\mathsf{B}} \mathit{var}\ \text{"y"} \,\rrparenthesis_{\Delta[\text{"x"} \mapsto X]}\right) \\
&= \forall^{\mathsf{B}} \mathbb{Z} \cdot (X \mapsto \mathit{var}\ X =^{\mathsf{B}} \mathit{var}\ Y)
\end{aligned}
$$

Remark

In the actual implementation, Δ is required to contain all free variables of the term t being abstracted: $\mathrm{dom}(\Delta) \supseteq \mathrm{fv}(t)$

Roadmap

- ☐ Syntax
- ☐ Type system
- ☐ Semantics

Roadmap

- ☑ Syntax
- ☐ Type system
- ☐ Semantics

Type system

We introduce a basic type system for B, based on the following types:

LEAN

```
inductive BType where
| int | bool | set : BType → BType | _ ×ᴮ _ : BType → BType → BType
```

Type system

We introduce a basic type system for B, based on the following types:

```
inductive BType where
| int | bool | set : BType → BType | _×ᴮ_ : BType → BType → BType
```

which can be transformed into terms:

```
def BType.toTerm : BType → Term
| int     ⇒ ℤ
| bool    ⇒ 𝔹
| set α   ⇒ 𝒫ᴮ α.toTerm
| α ×ᴮ β  ⇒ α.toTerm ×ᴮ β.toTerm
```

Typing rules

Type contexts are defined as follows:

LEAN
```
abbrev TypeContext :=
  AList fun _ : String ↦ BType
```

LEAN
```
abbrev PHOAS.TypeContext 𝒱 :=
  𝒱 → Option BType
```

Typing rules

Type contexts are defined as follows:

```lean
abbrev TypeContext :=
  AList fun _ : String ↦ BType
```

```lean
abbrev PHOAS.TypeContext 𝒱 :=
  𝒱 → Option BType
```

together with an abstraction function:

```lean
noncomputable def TypeContext.abstract {𝒱}
  (Δ : String → Option 𝒱) Γ : PHOAS.TypeContext 𝒱 := fun x : 𝒱 ↦
    if h : ∃ v ∈ Γ, Δ v = some x  then Γ.lookup (choose h)
    else none
```

Typing rules

Inductive typing predicate $\Gamma \vdash t : \tau$ for both concrete and abstract terms:

Typing rules

Inductive typing predicate $\Gamma \vdash t : \tau$ for both concrete and abstract terms:

$$\frac{\Gamma.lookup\ v = some\ \tau}{\Gamma \vdash var\ v : \tau}\ var_{\mathrm{I}}$$

$$\frac{\Gamma(v) = some\ \tau}{\Gamma \vdash var\ v : \tau}\ var_{\mathrm{I}}$$

Typing rules

Inductive typing predicate $\Gamma \vdash t : \tau$ for both concrete and abstract terms:

$$\frac{\Gamma.\mathit{lookup}\ v = \mathit{some}\ \tau}{\Gamma \vdash \mathit{var}\ v : \tau}\ \mathit{var}_{\mathrm{I}} \qquad \frac{\Gamma(v) = \mathit{some}\ \tau}{\Gamma \vdash \mathit{var}\ v : \tau}\ \mathit{var}_{\mathrm{I}}$$

$$\frac{\Gamma \vdash x : \alpha \qquad \Gamma \vdash y : \beta}{\Gamma \vdash x \mapsto^{\mathsf{B}} y : \alpha \times^{\mathsf{B}} \beta}\ \mathit{maplet}_{\mathrm{I}}$$

Typing rules

Inductive typing predicate $\Gamma \vdash t : \tau$ for both concrete and abstract terms:

$$\frac{\Gamma.\mathit{lookup}\ v = \mathit{some}\ \tau}{\Gamma \vdash \mathit{var}\ v : \tau}\ \mathit{var}_{\mathrm{I}} \qquad \frac{\Gamma(v) = \mathit{some}\ \tau}{\Gamma \vdash \mathit{var}\ v : \tau}\ \mathit{var}_{\mathrm{I}}$$

$$\frac{\Gamma \vdash x : \alpha \qquad \Gamma \vdash y : \beta}{\Gamma \vdash x \mapsto^{\mathsf{B}} y : \alpha \times^{\mathsf{B}} \beta}\ \mathit{maplet}_{\mathrm{I}}$$

$$\frac{\Gamma \vdash x : \alpha \qquad \Gamma \vdash S : \mathit{set}\ \alpha}{\Gamma \vdash x \in^{\mathsf{B}} S : \mathit{bool}}\ \mathit{mem}_{\mathrm{I}}$$

Typing rules

Inductive typing predicate $\Gamma \vdash t : \tau$ for both concrete and abstract terms:

$$\frac{\Gamma.\mathit{lookup}\ v = \mathit{some}\ \tau}{\Gamma \vdash \mathit{var}\ v : \tau}\ \mathit{var}_{\mathrm{I}} \qquad \frac{\Gamma(v) = \mathit{some}\ \tau}{\Gamma \vdash \mathit{var}\ v : \tau}\ \mathit{var}_{\mathrm{I}}$$

$$\frac{\Gamma \vdash x : \alpha \qquad \Gamma \vdash y : \beta}{\Gamma \vdash x \mapsto^{\mathsf{B}} y : \alpha \times^{\mathsf{B}} \beta}\ \mathit{maplet}_{\mathrm{I}}$$

$$\frac{\Gamma \vdash x : \alpha \qquad \Gamma \vdash S : \mathit{set}\ \alpha}{\Gamma \vdash x \in^{\mathsf{B}} S : \mathit{bool}}\ \mathit{mem}_{\mathrm{I}}$$

Similarly: $\mathit{add}_{\mathrm{I}}$, $\mathit{mul}_{\mathrm{I}}$, $\mathit{sub}_{\mathrm{I}}$, le_{I}, $\mathit{not}_{\mathrm{I}}$, eq_{I}, $\mathbb{Z}_{\mathrm{I}}$, $\mathbb{B}_{\mathrm{I}}$, $\mathit{mem}_{\mathrm{I}}$, $\mathit{pow}_{\mathrm{I}}$, $\mathit{cprod}_{\mathrm{I}}$, $\mathit{union}_{\mathrm{I}}$, $\mathit{inter}_{\mathrm{I}}$, $\mathit{card}_{\mathrm{I}}$, $\mathit{min}_{\mathrm{I}}$, $\mathit{max}_{\mathrm{I}}$.

Typing rules

Concrete lambda typing rule

Let $n \in \mathbb{N}^*$, Γ a concrete type context, $(v_i)_{i<n}$ concrete variables, $(\alpha_i)_{i<n}$ and ξ B types, $(D_i)_{i<n}$ and f concrete terms. Assume the following:

$$\forall i < n, v_i \notin \Gamma \,; \quad \forall i < n, \Gamma \vdash D_i : \mathit{set}\ \alpha_i \,; \quad \Gamma[v_i \coloneqq \alpha_i]_{i<n} \vdash f : \xi$$

Then, the following typing judgment holds:

$$\Gamma \vdash \lambda^{\mathrm{B}} v_1, \ldots, v_n \in^{\mathrm{B}} D_1 \times^{\mathrm{B}} \ldots \times^{\mathrm{B}} D_n \cdot f : \mathit{set}\ (\alpha_1 \times^{\mathrm{B}} \ldots \times^{\mathrm{B}} \alpha_n \times^{\mathrm{B}} \xi)$$

Typing rules

Abstract lambda typing rule

Let $n \in \mathbb{N}^*$, Γ an abstract type context, $(\alpha_i)_{i<n}$ and ξ B types, $(D_i)_{i<n}$ abstract terms and f an abstract motive. Assume the following:

$$\forall i < n,\ \Gamma \vdash D_i\ :\ set\ \alpha_i\ ; \quad \forall (v_i)_{i<n},\ \Gamma[v_i := \alpha_i]_{i<n} \vdash f\ ((v_i)_{i<n})\ :\ \xi$$

Then, the following typing judgment holds:

$$\Gamma \vdash \lambda^{\mathsf{B}} D_1 \times^{\mathsf{B}} \ldots \times^{\mathsf{B}} D_n \cdot f\ :\ set\ (\alpha_1 \times^{\mathsf{B}} \ldots \times^{\mathsf{B}} \alpha_n \times^{\mathsf{B}} \xi)$$

Auxiliary theorems

Auxiliary theorems

Theorem (Weakening)

Assume that $\Gamma \vdash t : \tau$. *Let* v *be a variable such that* $v \notin \Gamma$. *Then,*

$$\forall \alpha,\ \Gamma[v := \alpha] \vdash t : \tau$$

Auxiliary theorems

Theorem (Weakening)

Assume that $\Gamma \vdash t : \tau$. *Let* v *be a variable such that* $v \notin \Gamma$. *Then,*

$$\forall \alpha, \Gamma[v := \alpha] \vdash t : \tau$$

Theorem (Strengthening)

Assume that $\Gamma[v := \alpha] \vdash t : \tau$, *and* v *is not a free variable in* t. *Then,*

$$\Gamma \vdash t : \tau$$

Auxiliary theorems

Theorem (Weakening)

Assume that $\Gamma \vdash t : \tau$. Let v be a variable such that $v \notin \Gamma$. Then,

$$\forall \alpha,\ \Gamma[v := \alpha] \vdash t : \tau$$

Theorem (Strengthening)

Assume that $\Gamma[v := \alpha] \vdash t : \tau$, and v is not a free variable in t. Then,

$$\Gamma \vdash t : \tau$$

Theorem (Determinism)

$$\Gamma \vdash t : \tau \to \Gamma \vdash t : \sigma \to \tau = \sigma$$

Roadmap

- ☑ Syntax
- ☐ Type system
- ☐ Semantics

Roadmap

- ☑ Syntax
- ☑ Type system
- ☐ Semantics

Denotational semantics of B

ZFC

Interpreting the B syntax in ZFC set theory:

Denotational semantics of B

ZFC

Interpreting the B syntax in ZFC set theory: Lean provides a lacunary model of ZFC in Mathlib that we first had to extend.

Denotational semantics of B

ZFC

Interpreting the B syntax in ZFC set theory: Lean provides a lacunary model of ZFC in Mathlib that we first had to extend.

- `ZFSet`
- naturals, $\mathbb{N}$, `ZFNat` and arithmetics
- integers, $\mathbb{Z}$, `ZFInt` and arithmetics
- rationals, $\mathbb{Q}$, `ZFRat` and arithmetics
- reals, $\mathbb{R}$, `ZFReal` and arithmetics

Denotational semantics of B

ZFC

Interpreting the B syntax in ZFC set theory: Lean provides a lacunary model of ZFC in Mathlib that we first had to extend.

- `ZFSet`
- naturals, $\mathbb{N}$, `ZFNat` and arithmetics
- integers, $\mathbb{Z}$, `ZFInt` and arithmetics
- rationals, $\mathbb{Q}$, `ZFRat` and arithmetics
- reals, $\mathbb{R}$, `ZFReal` and arithmetics

Denotational semantics of B

ZFC

Interpreting the B syntax in ZFC set theory: Lean provides a lacunary model of ZFC in Mathlib that we first had to extend.

- `ZFSet`
- naturals, $\mathbb{N}$, `ZFNat` and arithmetics
- integers, $\mathbb{Z}$, `ZFInt` and arithmetics
- rationals, $\mathbb{Q}$, `ZFRat` and arithmetics
- reals, $\mathbb{R}$, `ZFReal` and arithmetics

Denotational semantics of B

ZFC

Interpreting the B syntax in ZFC set theory: Lean provides a lacunary model of ZFC in Mathlib that we first had to extend.

- `ZFSet`
- naturals, $\mathbb{N}$, `ZFNat` and arithmetics
- integers, $\mathbb{Z}$, `ZFInt` and arithmetics
- rationals, $\mathbb{Q}$, `ZFRat` and arithmetics
- reals, $\mathbb{R}$, `ZFReal` and arithmetics

Denotational semantics of B

We define a denotation function for abstract B terms:

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

Denotational semantics of B

We define a denotation function for abstract B terms:

$$[\![\cdot]\!]^{B} \quad : \quad \texttt{Term}\;\texttt{ZFSet} \rightarrow \texttt{ZFSet}$$

Denotational semantics of B

We define a denotation function for abstract B terms:

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

Denotational semantics of B

We define a denotation function for abstract B terms:

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

where

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

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

Denotational semantics of B

The whole purpose of the PHOAS is to pour semantics into the syntax:

Denotational semantics of B

The whole purpose of the PHOAS is to pour semantics into the syntax:

$$[\![var\ v]\!]^{B} \coloneqq v$$

Denotational semantics of B

The whole purpose of the PHOAS is to pour semantics into the syntax:

$$[\![var\ \langle V, \tau, \mathfrak{pf}_V^\tau \rangle]\!]^{B} \quad := \langle V, \tau, \mathfrak{pf}_V^\tau \rangle \qquad \text{where} \quad \mathfrak{pf}_V^\tau : V \in [\![\tau]\!]^{Z}$$

Denotational semantics of B

The whole purpose of the PHOAS is to pour semantics into the syntax:

$$\llbracket \mathit{var}\ \langle V, \tau, \mathfrak{pf}_V^{\tau} \rangle \rrbracket^{\mathsf{B}} := \langle V, \tau, \mathfrak{pf}_V^{\tau} \rangle \qquad \text{where} \quad \mathfrak{pf}_V^{\tau} : V \in \llbracket \tau \rrbracket^{\mathsf{Z}}$$

$$\llbracket \mathit{int}\ n \rrbracket^{\mathsf{B}} := \langle \mathtt{ofInt}\ n, \mathit{int}, \mathfrak{pf}_n \rangle \qquad \text{where} \quad \mathfrak{pf}_n : \mathtt{ofInt}\ n \in \mathbb{Z}^{\mathsf{Z}}$$

Denotational semantics of B

The whole purpose of the PHOAS is to pour semantics into the syntax:

$$\llbracket \mathit{var}\ \langle V, \tau, \mathfrak{pf}_V^{\tau} \rangle \rrbracket^{\mathsf{B}} \coloneqq \langle V, \tau, \mathfrak{pf}_V^{\tau} \rangle \quad \text{where} \quad \mathfrak{pf}_V^{\tau} : V \in \llbracket \tau \rrbracket^{\mathsf{Z}}$$

$$\llbracket \mathit{int}\ n \rrbracket^{\mathsf{B}} \coloneqq \langle \mathtt{ofInt}\ n, \mathit{int}, \mathfrak{pf}_n \rangle \quad \text{where} \quad \mathfrak{pf}_n : \mathtt{ofInt}\ n \in \mathbb{Z}^{\mathsf{Z}}$$

$$\llbracket \mathit{bool}\ b \rrbracket^{\mathsf{B}} \coloneqq \langle \mathtt{ofBool}\ b, \mathit{bool}, \mathfrak{pf}_b \rangle \quad \text{where} \quad \mathfrak{pf}_b : \mathtt{ofBool}\ b \in \mathbb{B}^{\mathsf{Z}}$$

Denotational semantics of B

Simple cases

Let's consider $\mapsto^{\mathsf{B}}$ only; other binary cases are similar.

Denotational semantics of B

Simple cases

Let's consider $\mapsto^{\mathrm{B}}$ only; other binary cases are similar. Let x and y be two abstract terms such that:

$$[\![x]\!]^{\mathrm{B}} = \langle X, \alpha, \mathfrak{pf}_X^{\alpha} \rangle \quad \text{and} \quad [\![y]\!]^{\mathrm{B}} = \langle Y, \beta, \mathfrak{pf}_Y^{\beta} \rangle$$

Denotational semantics of B

Simple cases

Let's consider $\mapsto^{\mathsf{B}}$ only; other binary cases are similar. Let x and y be two abstract terms such that:

$$[\![x]\!]^{\mathsf{B}} = \langle X, \alpha, \mathfrak{pf}_X^{\alpha} \rangle \quad \text{and} \quad [\![y]\!]^{\mathsf{B}} = \langle Y, \beta, \mathfrak{pf}_Y^{\beta} \rangle$$

The denotation is then constructed as follows:

$$[\![x \mapsto^{\mathsf{B}} y]\!]^{\mathsf{B}} := \langle (X, Y)^{\mathsf{Z}}, \alpha \times^{\mathsf{B}} \beta, \mathfrak{pf}_{(X,Y)^{\mathsf{Z}}}^{\alpha \times^{\mathsf{B}} \beta} \rangle$$

Denotational semantics of B

Simple cases

Let's consider $\mapsto^{\mathsf{B}}$ only; other binary cases are similar. Let x and y be two abstract terms such that:

$$[\![x]\!]^{\mathsf{B}} = \langle X, \alpha, \mathfrak{pf}_X^{\alpha} \rangle \quad \text{and} \quad [\![y]\!]^{\mathsf{B}} = \langle Y, \beta, \mathfrak{pf}_Y^{\beta} \rangle$$

The denotation is then constructed as follows:

$$[\![x \mapsto^{\mathsf{B}} y]\!]^{\mathsf{B}} := \langle (X, Y)^{\mathsf{Z}}, \alpha \times^{\mathsf{B}} \beta, \mathfrak{pf}_{(X,Y)^{\mathsf{Z}}}^{\alpha \times^{\mathsf{B}} \beta} \rangle$$

where

$$\mathfrak{pf}_{(X,Y)^{\mathsf{Z}}}^{\alpha \times^{\mathsf{B}} \beta} : \ (X, Y)^{\mathsf{Z}} \in [\![\alpha \times^{\mathsf{B}} \beta]\!]^{\mathsf{Z}}$$

Denotational semantics of B

Simple cases

Let's consider $\mapsto^{\mathsf{B}}$ only; other binary cases are similar. Let x and y be two abstract terms such that:

$$[\![x]\!]^{\mathsf{B}} = \langle X, \alpha, \mathfrak{pf}_X^{\alpha} \rangle \quad \text{and} \quad [\![y]\!]^{\mathsf{B}} = \langle Y, \beta, \mathfrak{pf}_Y^{\beta} \rangle$$

The denotation is then constructed as follows:

$$[\![x \mapsto^{\mathsf{B}} y]\!]^{\mathsf{B}} := \langle (X, Y)^{\mathsf{Z}}, \alpha \times^{\mathsf{B}} \beta, \mathfrak{pf}_{(X,Y)^{\mathsf{Z}}}^{\alpha \times^{\mathsf{B}} \beta} \rangle$$

where

$$\mathfrak{pf}_{(X,Y)^{\mathsf{Z}}}^{\alpha \times^{\mathsf{B}} \beta} \dashv (X, Y)^{\mathsf{Z}} \in [\![\alpha]\!]^{\mathsf{Z}} \times^{\mathsf{Z}} [\![\beta]\!]^{\mathsf{Z}}$$

Denotational semantics of B
Simple cases

Let's consider $\mapsto^{\mathsf{B}}$ only; other binary cases are similar. Let x and y be two abstract terms such that:

$$[\![x]\!]^{\mathsf{B}} = \langle X, \alpha, \mathfrak{pf}_X^{\alpha} \rangle \quad \text{and} \quad [\![y]\!]^{\mathsf{B}} = \langle Y, \beta, \mathfrak{pf}_Y^{\beta} \rangle$$

The denotation is then constructed as follows:

$$[\![x \mapsto^{\mathsf{B}} y]\!]^{\mathsf{B}} \coloneqq \langle (X, Y)^{\mathsf{Z}}, \alpha \times^{\mathsf{B}} \beta, \mathfrak{pf}_{(X,Y)^{\mathsf{Z}}}^{\alpha \times^{\mathsf{B}} \beta} \rangle$$

where

$$\mathfrak{pf}_{(X,Y)^{\mathsf{Z}}}^{\alpha \times^{\mathsf{B}} \beta} \dashv X \in [\![\alpha]\!]^{\mathsf{Z}} \wedge Y \in [\![\beta]\!]^{\mathsf{Z}}$$

Denotational semantics of B
Simple cases

Let's consider $\mapsto^{\mathsf{B}}$ only; other binary cases are similar. Let x and y be two abstract terms such that:

$$[\![x]\!]^{\mathsf{B}} = \langle X, \alpha, \mathfrak{pf}_X^{\alpha} \rangle \quad \text{and} \quad [\![y]\!]^{\mathsf{B}} = \langle Y, \beta, \mathfrak{pf}_Y^{\beta} \rangle$$

The denotation is then constructed as follows:

$$[\![x \mapsto^{\mathsf{B}} y]\!]^{\mathsf{B}} := \langle (X, Y)^{\mathsf{Z}}, \alpha \times^{\mathsf{B}} \beta, \mathfrak{pf}_{(X,Y)^{\mathsf{Z}}}^{\alpha \times^{\mathsf{B}} \beta} \rangle$$

where

$$\mathfrak{pf}_{(X,Y)^{\mathsf{Z}}}^{\alpha \times^{\mathsf{B}} \beta} \dashv \underbrace{X \in [\![\alpha]\!]^{\mathsf{Z}}}_{\mathfrak{pf}_X^{\alpha}} \wedge \underbrace{Y \in [\![\beta]\!]^{\mathsf{Z}}}_{\mathfrak{pf}_Y^{\beta}}$$

Denotational semantics of B

Forall quantifier

Binders are a little more tedious.

Denotational semantics of B

Forall quantifier

Binders are a little more tedious. Consider $D := \{e_1, \ldots, e_n\}$. Intuition:

$$\llbracket \forall^{B} D \cdot P \rrbracket^{B} = \llbracket P\ e_1 \rrbracket^{B} \wedge^{Z} \ldots \wedge^{Z} \llbracket P\ e_n \rrbracket^{B}$$

Denotational semantics of B

Forall quantifier

Binders are a little more tedious. Consider $D := \{e_1, \ldots, e_n\}$. Intuition:

$$\left[\!\left[\forall^{\mathsf{B}} D \cdot P\right]\!\right]^{\mathsf{B}} = [\![P\ e_1]\!]^{\mathsf{B}} \wedge^{\mathsf{Z}} \ldots \wedge^{\mathsf{Z}} [\![P\ e_n]\!]^{\mathsf{B}} = \bigwedge_{i=1}^{n}{}^{\mathsf{Z}} [\![P\ e_i]\!]^{\mathsf{B}}$$

Denotational semantics of B

Forall quantifier

Binders are a little more tedious. Consider $D := \{e_1, \ldots, e_n\}$. Intuition:

$$\left[\!\left[\forall^{\mathsf{B}} D \cdot P\right]\!\right]^{\mathsf{B}} = [\![P\ e_1]\!]^{\mathsf{B}} \wedge^{\mathsf{Z}} \ldots \wedge^{\mathsf{Z}} [\![P\ e_n]\!]^{\mathsf{B}} = \bigwedge_{i=1}^{n}{}^{\mathsf{Z}} [\![P\ e_i]\!]^{\mathsf{B}}$$

$$= \bigwedge_{x \in D}{}^{\mathsf{Z}} [\![P\ x]\!]^{\mathsf{B}}$$

Denotational semantics of B

Forall quantifier

Let D: `Term Dom` and P: $(\texttt{Fin}\ n \to \texttt{Dom}) \to \texttt{Term Dom}$.

Denotational semantics of B

Forall quantifier

Let D: `Term Dom` and P: $(\texttt{Fin}\ n \to \texttt{Dom}) \to$ `Term Dom`. Assume that

$$[\![D]\!]^{\mathsf{B}} = \langle \mathcal{D}, \mathit{set}\ \tau, \mathfrak{pf}_{\mathcal{D}}^{\mathit{set}\ \tau} \rangle \quad \text{and} \quad \tau = \tau_1 \times^{\mathsf{B}} \ldots \times^{\mathsf{B}} \tau_n$$

Denotational semantics of B
Forall quantifier

Let D: `Term Dom` and P: (`Fin` $n \to$ `Dom`) $\to$ `Term Dom`. Assume that

$$[\![D]\!]^{\mathsf{B}} = \langle \mathcal{D}, \mathit{set}\ \tau, \mathfrak{pf}_{\mathcal{D}}^{\mathit{set}\ \tau} \rangle \quad \text{and} \quad \tau = \tau_1 \times^{\mathsf{B}} \ldots \times^{\mathsf{B}} \tau_n$$

We define the following function, for any ZFC set z:

$$\mathscr{P}(z) := \begin{cases} P_z & \text{if } \exists (x_i)_{1 \le i \le n}, \begin{cases} z = (x_1, \ldots, x_n)^{\mathsf{Z}} \wedge z \in [\![\tau]\!]^{\mathsf{Z}} \\ \left[\!\!\left[P\ \left(\mathit{var}\ \langle x_i, \tau_i, \mathfrak{pf}_{x_i}^{\tau_i} \rangle\right)_{1 \le i \le n} \right]\!\!\right]^{\mathsf{B}} = \langle P_z, -, - \rangle \end{cases} \\ \bot^{\mathsf{Z}} & \text{otherwise} \end{cases}$$

Denotational semantics of B

Forall quantifier

Let $D\colon \texttt{Term Dom}$ and $P\colon (\texttt{Fin}\ n \to \texttt{Dom}) \to \texttt{Term Dom}$. Assume that

$$[\![D]\!]^{\mathsf{B}} = \langle \mathcal{D},\ \mathit{set}\ \tau,\ \mathfrak{pf}_{\mathcal{D}}^{\mathit{set}\ \tau} \rangle \quad \text{and} \quad \tau = \tau_1 \times^{\mathsf{B}} \ldots \times^{\mathsf{B}} \tau_n$$

We define the following function, for any ZFC set z:

$$\mathscr{P}(z) \coloneqq \begin{cases} P_z & \text{if } \exists (x_i)_{1\le i\le n}, \begin{cases} z = (x_1, \ldots, x_n)^{\mathsf{Z}} \land z \in [\![\tau]\!]^{\mathsf{Z}} \\ \left[\!\!\left[P\ \left(\mathit{var}\ \langle x_i,\ \tau_i,\ \mathfrak{pf}_{x_i}^{\tau_i} \rangle\right)_{1\le i\le n} \right]\!\!\right]^{\mathsf{B}} = \langle P_z, -, - \rangle \end{cases} \\ \bot^{\mathsf{Z}} & \text{otherwise} \end{cases}$$

At last, we define the following denotation:

$$[\![\forall^{\mathsf{B}} D \cdot P]\!]^{\mathsf{B}} \coloneqq \left\langle \bigwedge_{x\in\mathcal{D}}^{\mathsf{Z}} \mathscr{P}\ x,\ \mathit{bool},\ \mathfrak{pf}_{\forall^{\mathsf{B}} D\cdot P}^{\mathit{bool}} \right\rangle \quad \text{where} \quad \mathfrak{pf}_{\forall^{\mathsf{B}} D\cdot P}^{\mathit{bool}}\colon \bigwedge_{x\in\mathcal{D}}^{\mathsf{Z}} \mathscr{P}\ x \in [\![\mathit{bool}]\!]^{\mathsf{Z}}$$

Denotational semantics of B

Forall quantifier

We are not done yet!

Denotational semantics of B

Forall quantifier

We are not done yet! The following facts remain to be proved:

Denotational semantics of B

Forall quantifier

We are not done yet! The following facts remain to be proved:

$$\mathfrak{pf}^{bool}_{\forall^{\mathrm{B}} D \cdot P} : \quad \bigwedge_{x \in \mathcal{D}}^{z} \mathscr{P}\, x \in [\![bool]\!]^{z}$$

Denotational semantics of B

Forall quantifier

We are not done yet! The following facts remain to be proved:

$$\mathfrak{pf}^{bool}_{\forall^{B} D \cdot P} : \left(\bigwedge^{Z} \{ \mathscr{P}\ x,\ x \in \mathcal{D} \} \right) \in [\![bool]\!]^{Z}$$

Denotational semantics of B

Forall quantifier

We are not done yet! The following facts remain to be proved:

$$\mathfrak{pf}^{bool}_{\forall^{\mathrm{B}} D \cdot P} : \left(\bigwedge^{\mathrm{Z}} \{ \mathscr{P}\ x,\ x \in \mathcal{D} \} \right) \in \mathbb{B}$$

Denotational semantics of B

Forall quantifier

We are not done yet! The following facts remain to be proved:

$$\mathfrak{pf}^{bool}_{\forall^{\mathsf{B}} D \cdot P} : \left(\bigwedge^{\mathsf{Z}} \{ y \in \mathbb{B} \mid \exists x \in \mathcal{D},\, y = \mathscr{P}\, x \} \right) \in \mathbb{B}$$

Denotational semantics of B

Forall quantifier

We are not done yet! The following facts remain to be proved:

$$\mathfrak{pf}^{bool}_{\forall^{B} D\cdot P} : \left(\bigwedge^{z} \{y \in \mathbb{B} \mid \exists\, x \in \mathcal{D},\, y = \mathscr{P}\ x\}\right) \in \mathbb{B}$$

$$\forall\, z \in [\![\tau]\!]^{z},\, \exists (x_i)_{1 \le i \le n},\, z = (x_1, \ldots, x_n)^{z} \to \forall\, i, x_i \in [\![\tau_i]\!]^{z} \qquad (\mathfrak{pf}^{\tau_i}_{x_i})$$

Denotational semantics of B

Forall quantifier

We are not done yet! The following facts remain to be proved:

$$\mathfrak{pf}^{bool}_{\forall^{\mathsf{B}} D \cdot P} : \left(\bigwedge^{\mathsf{z}} \{ y \in \mathbb{B} \mid \exists x \in \mathcal{D}, y = \mathscr{P}\, x \} \right) \in \mathbb{B}$$

$$\forall z \in [\![\tau]\!]^{\mathsf{z}}, \exists (x_i)_{1 \le i \le n}, z = (x_1, \ldots, x_n)^{\mathsf{z}} \to \forall i, x_i \in [\![\tau_i]\!]^{\mathsf{z}} \qquad (\mathfrak{pf}^{\tau_i}_{x_i})$$

(easy proof)

Roadmap

- ☑ Syntax
- ☑ Type system
- ☐ Semantics

Roadmap

- ☑ Syntax
- ☑ Type system
- ☑ Semantics

Denotational semantics of B

We can now state properties about B terms!

Denotational semantics of B

We can now state properties about B terms!

Theorem (Type correctness of the denotation)

Assume that $\Gamma \vdash t : \tau$. *Then,*

$$[\![t]\!]^{\mathsf{B}} = \langle T, \sigma, \mathfrak{pf}_T^{\sigma} \rangle \to \sigma = \tau$$

Denotational semantics of B

We can now state properties about B terms!

Theorem (Type correctness of the denotation)

Assume that $\Gamma \vdash t : \tau$. *Then,*

$$[\![t]\!]^{\mathsf{B}} = \langle T, \sigma, \mathfrak{pf}^{\sigma}_{T} \rangle \rightarrow \sigma = \tau$$

Theorem (Partial correctness of the simplifier)

Assume that $\Gamma \vdash t : \tau$. *Let* Δ *be a renaming context. Then,*

$$[\![(\!| t |\!)_{\Delta}]\!]^{\mathsf{B}} = \langle T, \tau, \mathfrak{pf}^{\tau}_{T} \rangle \rightarrow [\![(\!| \mathsf{simplifier}(t) |\!)_{\Delta}]\!]^{\mathsf{B}} = \langle T, \tau, \mathfrak{pf}^{\tau}_{T} \rangle$$

Denotational semantics of B

We can now state properties about B terms!

Theorem (Type correctness of the denotation)

Assume that $\Gamma \vdash t : \tau$. *Then,*

$$[\![t]\!]^{\mathsf{B}} = \langle T, \sigma, \mathfrak{pf}_T^{\sigma} \rangle \to \sigma = \tau$$

Theorem (Partial correctness of the simplifier)

Assume that $\Gamma \vdash t : \tau$. *Let* Δ *be a renaming context. Then,*

$$[\![(\!|\ t\ |\!)_{\Delta}]\!]^{\mathsf{B}} = \langle T, \tau, \mathfrak{pf}_T^{\tau} \rangle \to [\![(\!|\ \mathsf{simplifier}(t)\ |\!)_{\Delta}]\!]^{\mathsf{B}} = \langle T, \tau, \mathfrak{pf}_T^{\tau} \rangle$$

Remark

Total correctness does not hold.

Conclusion

Contributions:

- **Higher-order encoding** leveraging recent advances in SMT solvers
- **Formal semantics** for subsets of B proof obligations and SMT-LIB

Current/future work:

- **Correctness** of the encoding