Towards Efficient and Verified Virtual Machines for Dynamic Languages

Martin Desharnais Stefan Brunthaler

Research Institute
Cyber Defence
Universität der Bundeswehr München

Certified Programs and Proofs
18–19 January 2021
Virtual, Denmark

The Success and Shortcoming of Virtual Machines

The Success and Shortcoming of Virtual Machines

- Performance > correctness & security

The Success and Shortcoming of Virtual Machines

- Performance > correctness & security
- Could the CompCert technique be applied to JIT compilers?

The Success and Shortcoming of Virtual Machines

- Performance > correctness & security
- Could the CompCert technique be applied to JIT compilers?
- Very big and complex code base

The Success and Shortcoming of Virtual Machines

- Performance > correctness & security
- Could the CompCert technique be applied to JIT compilers?
- Very big and complex code base
- Dynamic code generation

Optimization of Verified Interpreters is a More Realistic Paradigm

- Small & simple $\rightarrow$ easy formalization

Optimization of Verified Interpreters is a More Realistic Paradigm

- Small & simple $\rightarrow$ easy formalization
- Sufficient performance

Optimization of Verified Interpreters is a More Realistic Paradigm

- Small & simple $\rightarrow$ easy formalization
- Sufficient performance
- Balance between complexity and performance

Optimization of Verified Interpreters is a More Realistic Paradigm

- Small & simple $\rightarrow$ easy formalization
- Sufficient performance
- Balance between complexity and performance
- Promising experience with early implementation

Purely Interpretative Optimizations

Dyn Inca Ubx

Three languages

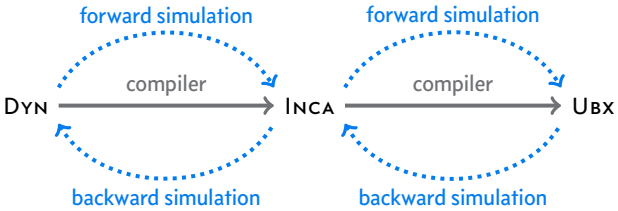
- **Dyn** Simple interpreter for dynamically typed languages
- **Inca** Type-based optimization (inline caching)
- **Ubx** Data-representation optimization (unboxing)

Purely Interpretative Optimizations

Three languages

- **Dyn** Simple interpreter for dynamically typed languages
- **Inca** Type-based optimization (inline caching)
- **Ubx** Data-representation optimization (unboxing)

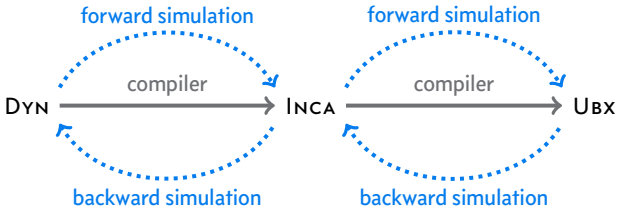
Purely Interpretative Optimizations



Three languages

- DYN Simple interpreter for dynamically typed languages
- INCA Type-based optimization (inline caching)
- UBX Data-representation optimization (unboxing)

Purely Interpretative Optimizations



Three languages

- DYN Simple interpreter for dynamically typed languages
- INCA Type-based optimization (inline caching)
- UBX Data-representation optimization (unboxing)

Mechanized formalization

- Isabelle/HOL
- 4800 lines + 1000 background
- Extensive use of locales
- Realistic small-step operational semantics

Dyn: Simple Interpreter for Dynamically Typed Languages

Features

- Operand stack
- Dynamic memory
- *n*-ary operations
- Conditional jumps
- Recursive function calls

Values

```
data dynamic =
  DI int |
  DF float |
  …
```

Dyn: Execution of Small Example Program

Program

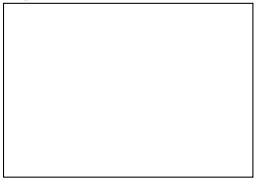
```
Celsius = (Fahrenheit - 32) * 5/9
Load fahrenheit
Push (DI 32)
Op Sub
Push (DI 5)
Op Mul
Push (DI 9)
Op Div
Store celsius
```

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
→  Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

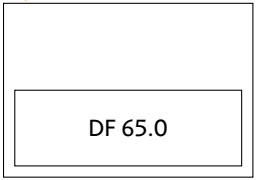
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ $\cdot$}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
→  Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

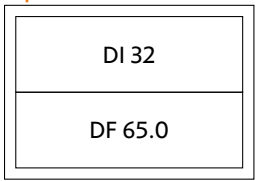
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ ·}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
→  Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

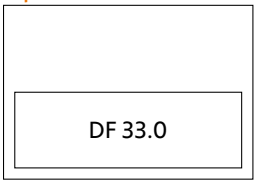
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ $\cdot$}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
→  Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

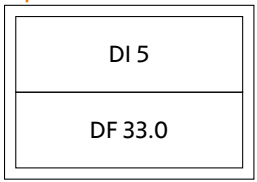
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ $\cdot$}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
→  Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

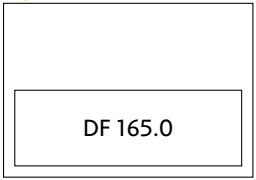
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ $\cdot$}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
→  Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

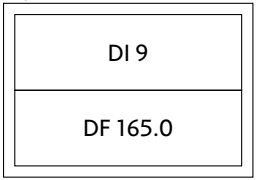
{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
→  Op Div
   Store celsius
```

Operand stack



Dynamic memory

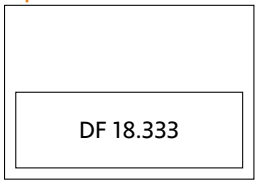
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ $\cdot$}

DYN: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
→  Store celsius
```

Operand stack



Dynamic memory

{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ ·}

Dyn: Execution of Small Example Program

Program

```
Celsius = (Fahrenheit - 32) * 5/9
Load fahrenheit
Push (DI 32)
Op Sub
Push (DI 5)
Op Mul
Push (DI 9)
Op Div
Store celsius
```

Operand stack

Dynamic memory

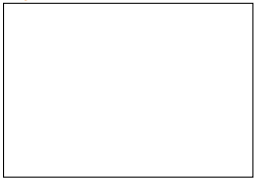
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ DF 18.333}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
→  Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

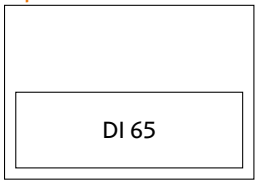
{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ ·}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
→  Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

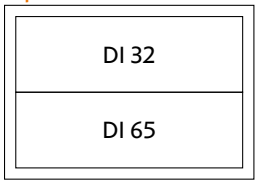
{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ ·}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
→  Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

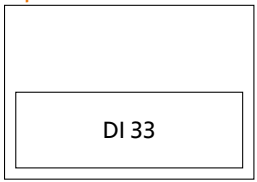
{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ ·}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
→  Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

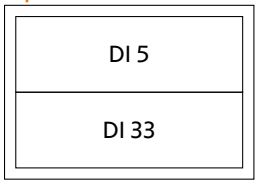
{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ ·}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
→  Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

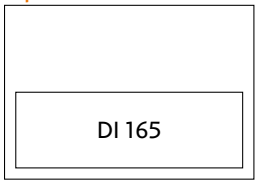
{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ ·}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
→  Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

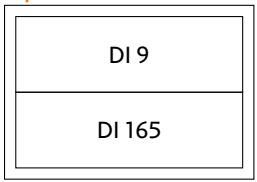
{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ ·}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
→  Op Div
   Store celsius
```

Operand stack



Dynamic memory

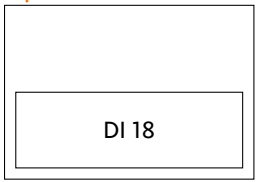
{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ $\cdot$}

Dyn: Execution of Small Example Program

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
→  Store celsius
```

Operand stack



Dynamic memory

{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ ·}

Dyn: Execution of Small Example Program

Program

```
Celsius = (Fahrenheit - 32) * 5/9
Load fahrenheit
Push (DI 32)
Op Sub
Push (DI 5)
Op Mul
Push (DI 9)
Op Div
Store celsius
```

Operand stack

Dynamic memory

{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ DI 18}

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)` → 6 type checks

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)` → 6 type checks

Example with inlining

```
fun eval_inl Div_float×int (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | eval_inl Div_int×int   (DI x) (DI y) = DI (Int.div x y)
  | eval_inl Div_float×float (DF x) (DF y) = DF (Float.div x y)
  | eval_inl Div_int×int   (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | …
```

e.g. `eval` $\mathtt{Div}_{\mathtt{float}\times\mathtt{int}}$ `(DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)` → 6 type checks

Example with inlining

```
fun eval_inl Div_float×int (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | eval_inl Div_int×int   (DI x) (DI y) = DI (Int.div x y)
  | eval_inl Div_float×float (DF x) (DF y) = DF (Float.div x y)
  | eval_inl Div_int×int   (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | …
```

e.g. `eval` $\mathtt{Div}_{\mathtt{float}\times\mathtt{int}}$ `(DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

e.g. `eval Div (DF 165.0) (DI 9)` → 6 type checks

Example with inlining

```
fun eval_inl Div_float×int (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | eval_inl Div_int×int   (DI x) (DI y) = DI (Int.div x y)
  | eval_inl Div_float×float (DF x) (DF y) = DF (Float.div x y)
  | eval_inl Div_int×int   (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | …
```

e.g. `eval` $\mathtt{Div}_{\mathtt{float}\times\mathtt{int}}$ `(DF 165.0) (DI 9)`

Inlining Reduces the Number of Type Checks in Operations

Example without inlining

```
fun eval Div (DI x) (DI y) = DI (Int.div x y)
  | eval Div (DF x) (DF y) = DF (Float.div x y)
  | eval Div (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | eval Div (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | …
```

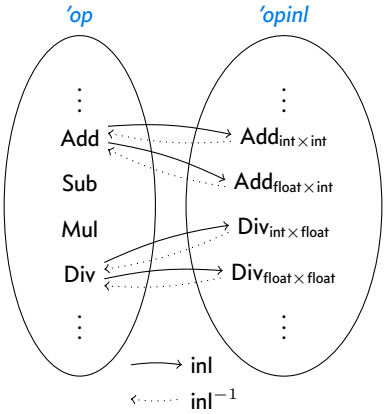
e.g. `eval Div (DF 165.0) (DI 9)` → 6 type checks

Example with inlining

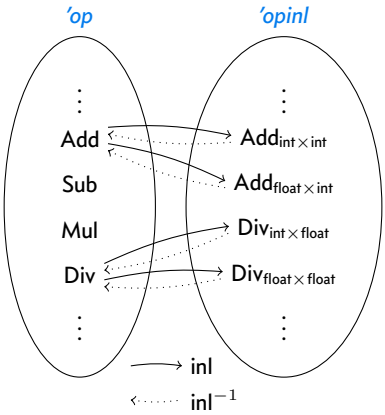
```
fun eval_inl Div_float×int (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | eval_inl Div_int×int   (DI x) (DI y) = DI (Int.div x y)
  | eval_inl Div_float×float (DF x) (DF y) = DF (Float.div x y)
  | eval_inl Div_int×int   (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | …
```

e.g. `eval` $\text{Div}_{\text{float}\times\text{int}}$ `(DF 165.0) (DI 9)` → 2 type checks

The Sets of Regular and Inlined Operations

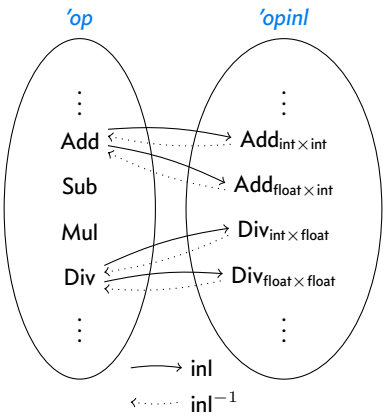


The Sets of Regular and Inlined Operations



locale operations =
  fixes
    arith :: *'op* ⇒ nat **and**
    eval :: *'op* ⇒ dynamic list ⇒ dynamic

The Sets of Regular and Inlined Operations



locale operations =
  fixes
    arith :: *'op* ⇒ nat **and**
    eval :: *'op* ⇒ dynamic list ⇒ dynamic

locale operations-inl = operations +
  fixes
    eval-inl :: *'opinl* ⇒ dynamic list ⇒ dynamic **and**
    inl :: *'op* ⇒ dynamic list ⇒ *'opinl* option **and**
    inl^{-1} :: *'opinl* ⇒ *'op* **and**
    …
  assumes
    inl *op args* = Some op_{inl} ⟹ eval *op* = eval-inl op_{inl} **and**
    inl *op args* = Some op_{inl} ⟹ inl^{-1} op_{inl} = *op* **and**
    …

Inca: Dyn Plus Inline Caching

+1 instruction for inlined operations

Dyn rules

$\vdots$

$\rightarrow_{\text{Dyn}}$-Op

$\vdots$

Inca rules

$\vdots$

$\rightarrow_{\text{Inca}}$-Op-Inl

$\rightarrow_{\text{Inca}}$-Op-No-Inl

$\rightarrow_{\text{Inca}}$-Op-Cache-Hit

$\rightarrow_{\text{Inca}}$-Op-Cache-Miss

$\vdots$

INCA: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
→  Load fahrenheit
   Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack

Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Language	Pass	Type checks
DYN	all	0
INCA	1^{st}	0
INCA	n^{th}	

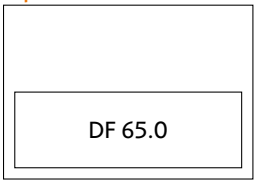
INCA: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
→  Push (DI 32)
   Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Language	Pass	Type checks
DYN	all	0
INCA	1^{st}	0
INCA	n^{th}	

Operand stack



Dynamic memory

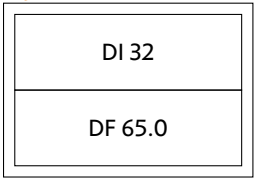
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ ·}

Inca: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
→  Op Sub
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Language	Pass	Type checks
Dyn	all	0
Inca	1^{st}	0
Inca	n^{th}	

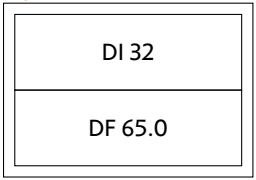
INCA: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
→  OpInl Sub_float×int
   Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Language	Pass	Type checks
DYN	all	6
INCA	1^{st}	8
INCA	n^{th}	

Operand stack



Dynamic memory

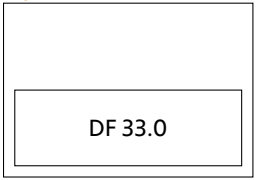
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ ·}

INCA: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
→  Push (DI 5)
   Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Language	Pass	Type checks
DYN	all	6
INCA	1^{st}	8
INCA	n^{th}	

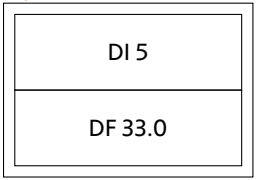
INCA: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
→  Op Mul
   Push (DI 9)
   Op Div
   Store celsius
```

Language	Pass	Type checks
DYN	all	6
INCA	1^{st}	8
INCA	n^{th}	

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

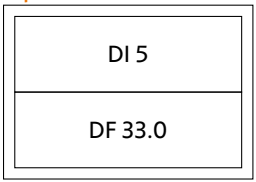
Inca: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
→  OpInl Mul_float×int
   Push (DI 9)
   Op Div
   Store celsius
```

Language	Pass	Type checks
Dyn	all	12
Inca	1^{st}	16
Inca	n^{th}	

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

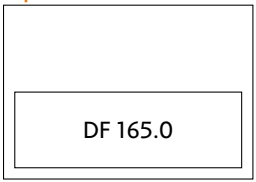
INCA: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
→  Push (DI 9)
   Op Div
   Store celsius
```

Language	Pass	Type checks
DYN	all	12
INCA	1^{st}	16
INCA	n^{th}	

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

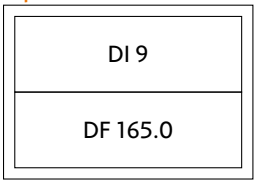
Inca: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
→  Op Div
   Store celsius
```

Language	Pass	Type checks
Dyn	all	12
Inca	1^{st}	16
Inca	n^{th}	

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

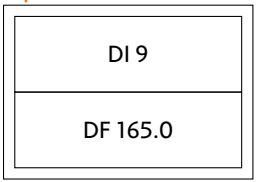
Inca: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
→  OpInl Div_float×int
   Store celsius
```

Language	Pass	Type checks
Dyn	all	18
Inca	1^{st}	24
Inca	n^{th}	

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

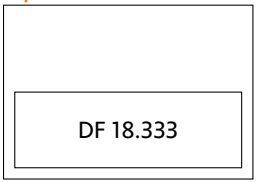
INCA: Execution of Small Example Program

Program: 1st pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
→  Store celsius
```

Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Inca: Execution of Small Example Program

Program: 1st pass

```
Celsius = (Fahrenheit - 32) * 5/9
Load fahrenheit
Push (DI 32)
OpInl Sub_float×int
Push (DI 5)
OpInl Mul_float×int
Push (DI 9)
OpInl Div_float×int
Store celsius
```

Language	Pass	Type checks
Dyn	all	18
Inca	1^{st}	24
Inca	n^{th}	

Operand stack

Dynamic memory

{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ DF 18.333}

INCA: Execution of Small Example Program

Program: n^{th} pass

```
   Celsius = (Fahrenheit - 32) * 5/9
→  Load fahrenheit
   Push (DI 32)
   OpInl Sub_{float×int}
   Push (DI 5)
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack

Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

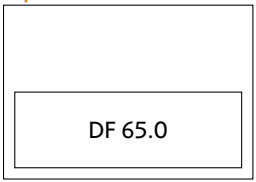
Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	0

INCA: Execution of Small Example Program

Program: n^{th} pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
→  Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	0

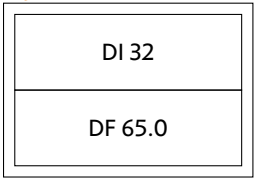
INCA: Execution of Small Example Program

Program: n^{th} pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
→  OpInl Sub_{float×int}
   Push (DI 5)
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	0

Operand stack



Dynamic memory

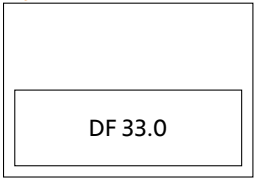
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ ·}

INCA: Execution of Small Example Program

Program: n^{th} pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
→  Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	4

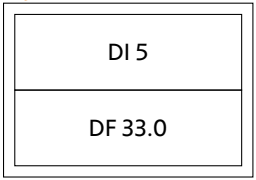
INCA: Execution of Small Example Program

Program: n^{th} pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
→  OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	4

Operand stack



Dynamic memory

{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ ·}

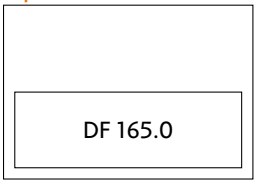
INCA: Execution of Small Example Program

Program: n^{th} pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
→  Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	8

Operand stack



Dynamic memory

{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ ·}

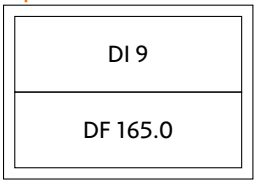
INCA: Execution of Small Example Program

Program: n^{th} pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
→  OpInl Div_float×int
   Store celsius
```

Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	8

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

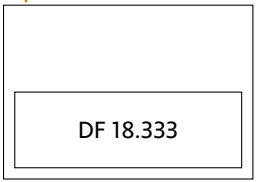
INCA: Execution of Small Example Program

Program: n^{th} pass

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
→  Store celsius
```

Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	12

Operand stack



Dynamic memory

{fahrenheit ↦ DF 65.0, celsius ↦ ·}

INCA: Execution of Small Example Program

Program: n^{th} pass

```
Celsius = (Fahrenheit - 32) * 5/9
Load fahrenheit
Push (DI 32)
OpInl Sub_float×int
Push (DI 5)
OpInl Mul_float×int
Push (DI 9)
OpInl Div_float×int
Store celsius
```

Language	Pass	Type checks
DYN	all	18
INCA	1^{st}	24
INCA	n^{th}	12

Operand stack

Dynamic memory

{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ DF 18.333}

Inca: Compiler is Sound and Complete

The compiler $\mathcal{C}$ trivialy lifts instructions from Dyn to Inca.

Soundness

There exists a bisimulation $\sim$, such that $p \sim \mathcal{C}\, p$.

Completeness

$\mathcal{C}$ is a total function.

Unboxing Completely Removes Type Checks in Operations

Example without unboxing

```
fun eval_inl Div_{float×int}   (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | eval_inl Div_{int×int}     (DI x) (DI y) = DI (Int.div x y)
  | eval_inl Div_{float×float} (DF x) (DF y) = DF (Float.div x y)
  | eval_inl Div_{int×int}     (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | …
```

Unboxing Completely Removes Type Checks in Operations

Example without unboxing

```
fun eval_inl Div_float×int   (DF x) (DI y) = DF (Float.div x (Float.from_int y))
  | eval_inl Div_int×int     (DI x) (DI y) = DI (Int.div x y)
  | eval_inl Div_float×float (DF x) (DF y) = DF (Float.div x y)
  | eval_inl Div_int×int     (DI x) (DF y) = DF (Float.div (Float.from_int x) y)
  | …
```

Example with unboxing

```
fun eval_ubx Div_float×int x y = Float.div x (Float.from_int y)
  | …
```

UBX: INCA Plus Unboxing

+n instructions for pushing unboxed values
+2 instructions for loading/storing unboxed values
+1 instructions for unboxed operations

INCA rules

⋮

$\rightarrow_{\text{INCA}}$-Push

$\rightarrow_{\text{INCA}}$-Load

$\rightarrow_{\text{INCA}}$-Store

⋮

UBX rules

⋮

$\rightarrow_{\text{UBX}}$-Push

$\rightarrow_{\text{UBX}}$-Push-Int

$\rightarrow_{\text{UBX}}$-Push-Float

$\rightarrow_{\text{UBX}}$-Load

$\rightarrow_{\text{UBX}}$-Load-Ubx-Hit

$\rightarrow_{\text{UBX}}$-Load-Ubx-Miss

$\rightarrow_{\text{UBX}}$-Op-Ubx

$\rightarrow_{\text{UBX}}$-Store

$\rightarrow_{\text{UBX}}$-Store-Ubx

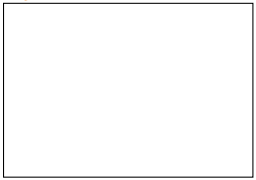
⋮

Ubx: Execution of Small Example Program

Program with **cache hit**

```
   Celsius = (Fahrenheit - 32) * 5/9
→  LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
   PushInt 5
   OpUbx Mul_{float×int}
   PushInt 9
   OpUbx Div_{float×int}
   StoreUbx float celsius
```

Operand stack



Dynamic memory

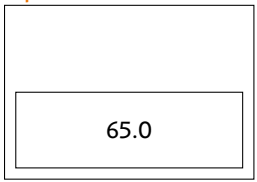
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ ·}

Ubx: Execution of Small Example Program

Program with **cache hit**

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
→  PushInt 32
   OpUbx Sub_float×int
   PushInt 5
   OpUbx Mul_float×int
   PushInt 9
   OpUbx Div_float×int
   StoreUbx float celsius
```

Operand stack



Dynamic memory

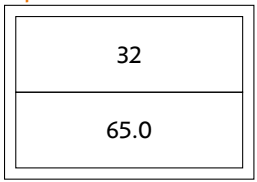
{fahrenheit ↦ DF 65.0, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache hit**

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
→  OpUbx Sub_float×int
   PushInt 5
   OpUbx Mul_float×int
   PushInt 9
   OpUbx Div_float×int
   StoreUbx float celsius
```

Operand stack



Dynamic memory

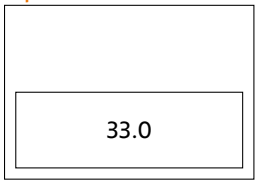
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ ·}

UBX: Execution of Small Example Program

Program with **cache hit**

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_float×int
→  PushInt 5
   OpUbx Mul_float×int
   PushInt 9
   OpUbx Div_float×int
   StoreUbx float celsius
```

Operand stack



Dynamic memory

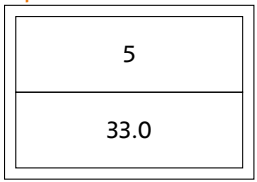
{fahrenheit ↦ DF 65.0, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache hit**

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_float×int
   PushInt 5
→  OpUbx Mul_float×int
   PushInt 9
   OpUbx Div_float×int
   StoreUbx float celsius
```

Operand stack



Dynamic memory

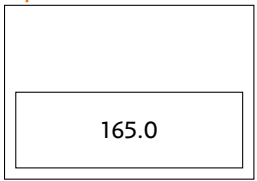
{fahrenheit ↦ DF 65.0, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache hit**

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_float×int
   PushInt 5
   OpUbx Mul_float×int
→  PushInt 9
   OpUbx Div_float×int
   StoreUbx float celsius
```

Operand stack



Dynamic memory

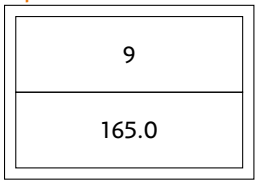
{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Ubx: Execution of Small Example Program

Program with **cache hit**

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_float×int
   PushInt 5
   OpUbx Mul_float×int
   PushInt 9
→  OpUbx Div_float×int
   StoreUbx float celsius
```

Operand stack



Dynamic memory

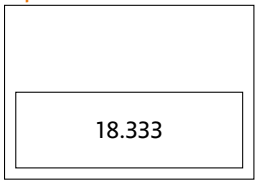
{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Ubx: Execution of Small Example Program

Program with **cache hit**

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_float×int
   PushInt 5
   OpUbx Mul_float×int
   PushInt 9
   OpUbx Div_float×int
→  StoreUbx float celsius
```

Operand stack



Dynamic memory

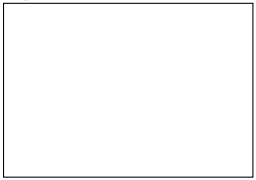
{fahrenheit ↦ DF 65.0, celsius ↦ ·}

Ubx: Execution of Small Example Program

Program with **cache hit**

```
Celsius = (Fahrenheit - 32) * 5/9
LoadUbx float fahrenheit
PushInt 32
OpUbx Sub_float×int
PushInt 5
OpUbx Mul_float×int
PushInt 9
OpUbx Div_float×int
StoreUbx float celsius
```

Operand stack



Dynamic memory

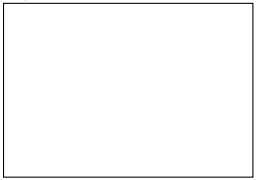
{fahrenheit $\mapsto$ DF 65.0, celsius $\mapsto$ DF 18.333}

UBX: Execution of Small Example Program

Program with **cache miss**

```
   Celsius = (Fahrenheit - 32) * 5/9
→  LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_float×int
   PushInt 5
   OpUbx Mul_float×int
   PushInt 9
   OpUbx Div_float×int
   StoreUbx float celsius
```

Operand stack



Dynamic memory

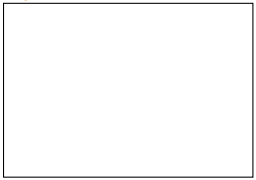
{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ ·}

UBX: Execution of Small Example Program

Program with **cache miss**

```
   Celsius = (Fahrenheit - 32) * 5/9
→  Load fahrenheit
   Push (DI 32)
   OpInl Sub_{float×int}
   Push (DI 5)
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Dynamic memory

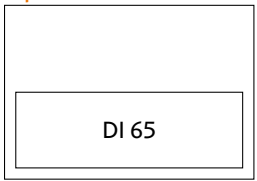
{fahrenheit ↦ DI 65, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache miss**

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
→  Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Operand stack



Dynamic memory

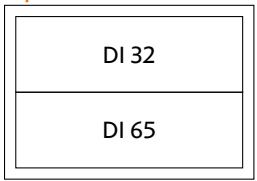
{fahrenheit ↦ DI 65, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache miss**

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
→  OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Operand stack



Dynamic memory

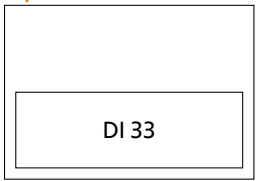
{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ ·}

UBX: Execution of Small Example Program

Program with **cache miss**

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
→  Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Operand stack



Dynamic memory

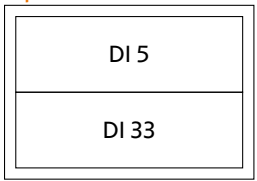
{fahrenheit ↦ DI 65, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache miss**

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
→  OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Operand stack



Dynamic memory

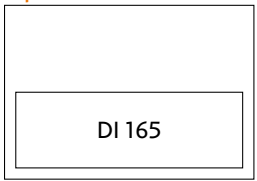
{fahrenheit ↦ DI 65, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache miss**

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
→  Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Operand stack



Dynamic memory

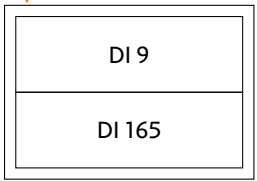
{fahrenheit ↦ DI 65, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache miss**

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
→  OpInl Div_float×int
   Store celsius
```

Operand stack



Dynamic memory

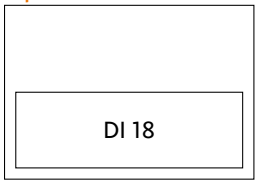
{fahrenheit ↦ DI 65, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache miss**

```
   Celsius = (Fahrenheit - 32) * 5/9
   Load fahrenheit
   Push (DI 32)
   OpInl Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
→  Store celsius
```

Operand stack



Dynamic memory

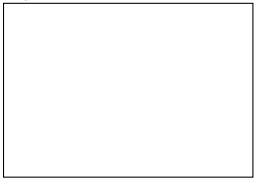
{fahrenheit ↦ DI 65, celsius ↦ ·}

UBX: Execution of Small Example Program

Program with **cache miss**

```
Celsius = (Fahrenheit - 32) * 5/9
Load fahrenheit
Push (DI 32)
OpInl Sub_float×int
Push (DI 5)
OpInl Mul_float×int
Push (DI 9)
OpInl Div_float×int
Store celsius
```

Operand stack



Dynamic memory

{fahrenheit $\mapsto$ DI 65, celsius $\mapsto$ DI 18}

Ubx: Static Optimization

Abstract interpretation

Ubx: Static Optimization

Abstract interpretation

Static type information

- Constants
- Function parameters
- Static analysis

Oracle

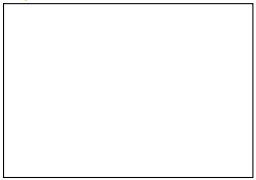
What is the exected type when loading a value from memory?

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
→  Load fahrenheit
   Push (DI 32)
   OpInl Sub_{float×int}
   Push (DI 5)
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

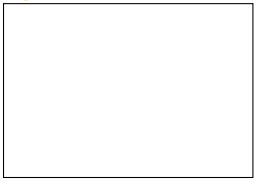
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
→  LoadUbx float fahrenheit
   Push (DI 32)
   OpInl Sub_{float×int}
   Push (DI 5)
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

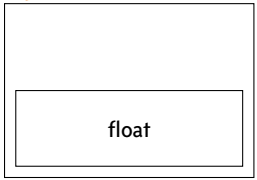
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
→  Push (DI 32)
   OpInl Sub_{float×int}
   Push (DI 5)
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

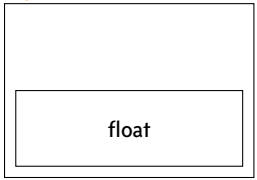
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
→  PushInt 32
   OpInl Sub_{float×int}
   Push (DI 5)
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

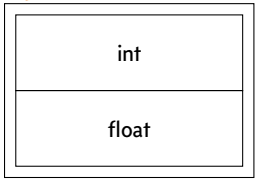
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
→  OpInl Sub_{float×int}
   Push (DI 5)
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

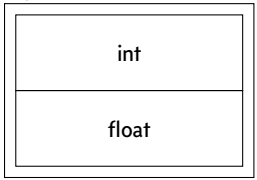
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
→  OpUbx Sub_float×int
   Push (DI 5)
   OpInl Mul_float×int
   Push (DI 9)
   OpInl Div_float×int
   Store celsius
```

Operand stack



Oracle

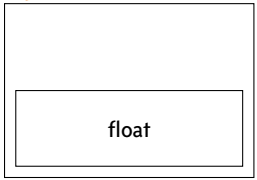
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
→  Push (DI 5)
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

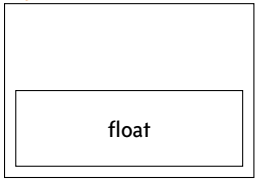
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
→  PushInt 5
   OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

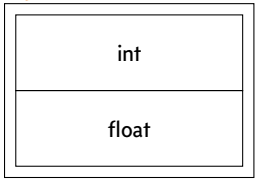
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
   PushInt 5
→  OpInl Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

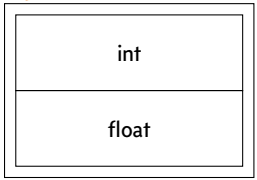
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
   PushInt 5
→  OpUbx Mul_{float×int}
   Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

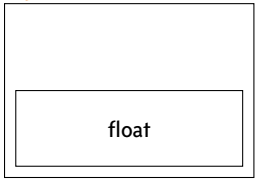
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
   PushInt 5
   OpUbx Mul_{float×int}
→  Push (DI 9)
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

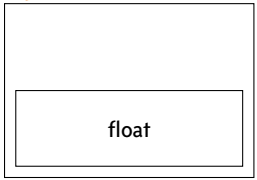
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
   PushInt 5
   OpUbx Mul_{float×int}
→  PushInt 9
   OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

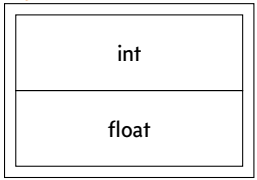
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
   PushInt 5
   OpUbx Mul_{float×int}
   PushInt 9
→  OpInl Div_{float×int}
   Store celsius
```

Operand stack



Oracle

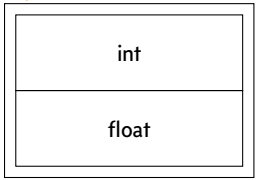
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
   PushInt 5
   OpUbx Mul_{float×int}
   PushInt 9
→  OpUbx Div_{float×int}
   Store celsius
```

Operand stack



Oracle

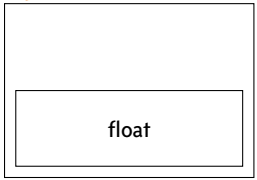
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
   PushInt 5
   OpUbx Mul_{float×int}
   PushInt 9
   OpUbx Div_{float×int}
→  Store celsius
```

Operand stack



Oracle

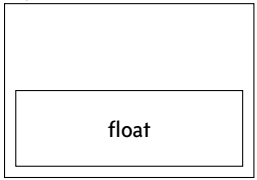
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
   Celsius = (Fahrenheit - 32) * 5/9
   LoadUbx float fahrenheit
   PushInt 32
   OpUbx Sub_{float×int}
   PushInt 5
   OpUbx Mul_{float×int}
   PushInt 9
   OpUbx Div_{float×int}
→  StoreUbx float celsius
```

Operand stack



Oracle

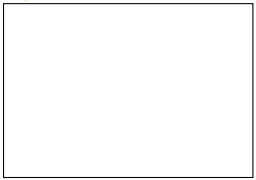
$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \cdot\}$

Ubx: Example of Static Optimization

Program

```
Celsius = (Fahrenheit - 32) * 5/9
LoadUbx float fahrenheit
PushInt 32
OpUbx Sub_{float×int}
PushInt 5
OpUbx Mul_{float×int}
PushInt 9
OpUbx Div_{float×int}
StoreUbx float celsius
```

Operand stack



Oracle

$\{\text{fahrenheit} \mapsto \mathit{float}, \mathit{celsius} \mapsto \mathit{float}\}$

UBX: Compiler is Sound and Almost Complete

The compiler $\mathcal{C}$

1. lifts instructions from INCA to UBX;
2. performs a single-pass, optimizing abstract analysis;
3. falls back to the unoptimized version if optimization failed.

Soundness

If p is "well-typed", there exists a bisimulation $\sim$, such that $p \sim \mathcal{C}\, p$.

Completeness

$\mathcal{C}$ is a partial function.
It is complete only for "well-typed" programs.

Summary

Specification of purely interpretative optimizations

Dyn: *baseline*

```
Load fahrenheit
Push (DI 32)
Op Sub
Push (DI 5)
Op Mul
Push (DI 9)
Op Div
Store celsius
```

Inca: *inline caching*

```
Load fahrenheit
Push (DI 32)
OpInl Sub_float×int
Push (DI 5)
OpInl Mul_float×int
Push (DI 9)
OpInl Div_float×int
Store celsius
```

Ubx: *unboxing*

```
LoadUbx float fahrenheit
PushInt 32
OpUbx Sub_float×int
PushInt 5
OpUbx Mul_float×int
PushInt 9
OpUbx Div_float×int
StoreUbx float celsius
```

Summary

Specification of purely interpretative optimizations

Dyn: *baseline*
```
Load fahrenheit
Push (DI 32)
Op Sub
Push (DI 5)
Op Mul
Push (DI 9)
Op Div
Store celsius
```

Inca: *inline caching*
```
Load fahrenheit
Push (DI 32)
OpInl Sub_float×int
Push (DI 5)
OpInl Mul_float×int
Push (DI 9)
OpInl Div_float×int
Store celsius
```

Ubx: *unboxing*
```
LoadUbx float fahrenheit
PushInt 32
OpUbx Sub_float×int
PushInt 5
OpUbx Mul_float×int
PushInt 9
OpUbx Div_float×int
StoreUbx float celsius
```

Complete mechanical formalization

- Realistic small-step operational semantics
- Multiple details kept abstract with locales
- Available in the *Archive of Formal Proofs*
  `https://isa-afp.org/entries/Interpreter_Optimizations.html`

Summary

Specification of purely interpretative optimizations

Dyn: *baseline*

```
Load fahrenheit
Push (DI 32)
Op Sub
Push (DI 5)
Op Mul
Push (DI 9)
Op Div
Store celsius
```

Inca: *inline caching*

```
Load fahrenheit
Push (DI 32)
OpInl Sub_float×int
Push (DI 5)
OpInl Mul_float×int
Push (DI 9)
OpInl Div_float×int
Store celsius
```

Ubx: *unboxing*

```
LoadUbx float fahrenheit
PushInt 32
OpUbx Sub_float×int
PushInt 5
OpUbx Mul_float×int
PushInt 9
OpUbx Div_float×int
StoreUbx float celsius
```

Complete mechanical formalization

- Realistic small-step operational semantics
- Multiple details kept abstract with locales
- Available in the *Archive of Formal Proofs*
  `https://isa-afp.org/entries/Interpreter_Optimizations.html`

Thank you