# Melocoton

## A Program Logic for Verified Interoperability Between OCaml and C

Armaël Guéneau, Johannes Hostert, Simon Spies,

Michael Sammler, Lars Birkedal, Derek Dreyer

OOPSLA 2023, Cascais

27 October, 2023

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

# Multi-Language Programs Are Everywhere


Python

C

Fortran


C++

Rust

JavaScript


C

Bindings for:

- Rust
- Python
- OCaml
- Go
- …

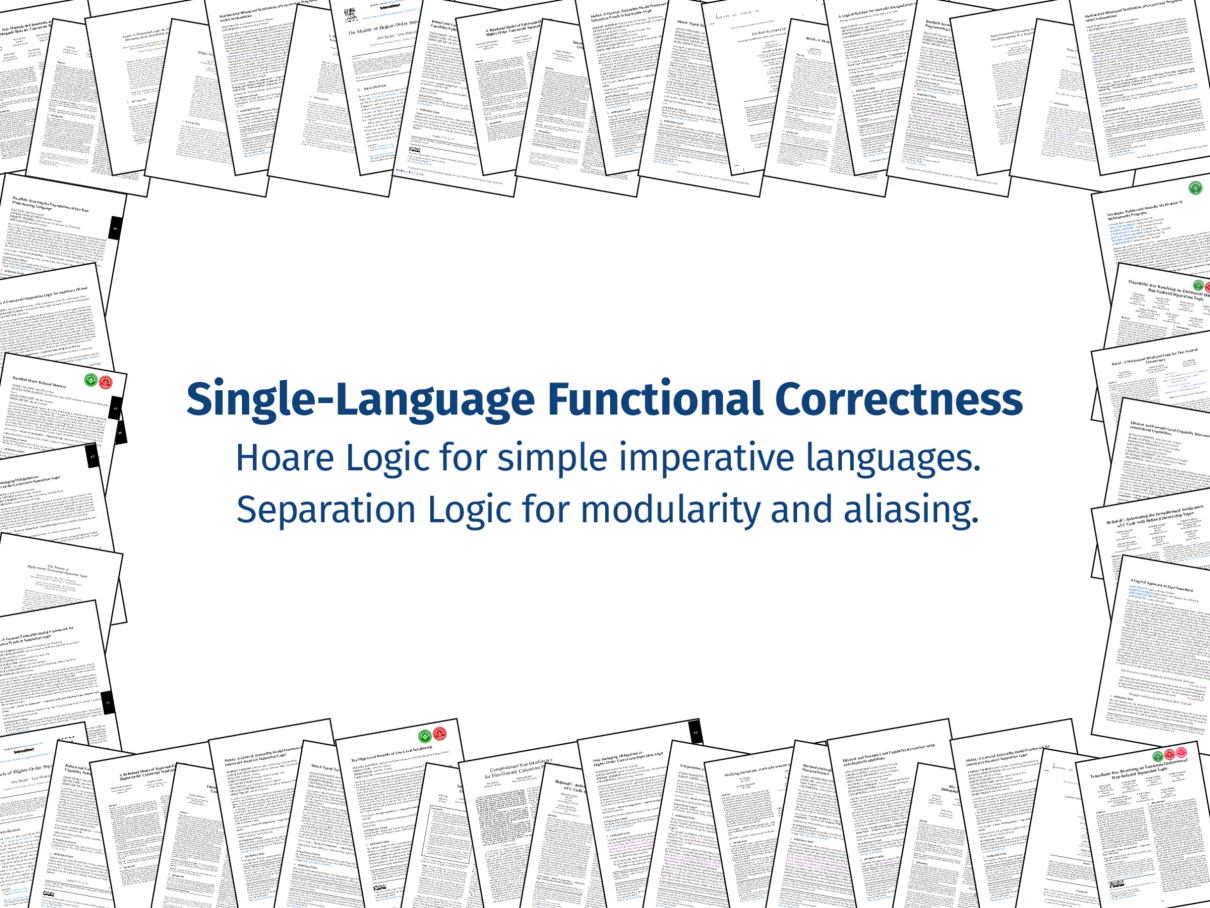
# The Goal: Verifying Multi-Language Programs

How do we

**verify functional correctness**

of programs written in

**different languages**?



# Single-Language Functional Correctness

Hoare Logic for simple imperative languages.
Separation Logic for modularity and aliasing.

# Multi-Language Functional Correctness

# Multi-Language Functional Correctness

Existing work on Semantics and Logical Relations.
How do we prove functional correctness of
individual, potentially unsafe programs?

# A Multi-Language Program in OCaml and C

# A Multi-Language Program in OCaml and C

## C business logic

```
void hash_ptr(int * x) {
    // Implemented in OpenSSL
    // tedious to port to OCaml
}
```

# A Multi-Language Program in OCaml and C

**OCaml** business logic

```
let main () =
  let r = ref 42 in
  hash_ref r; (*written in C*)
  print_int !r
```

**C** business logic

```
void hash_ptr(int * x) {
    // Implemented in OpenSSL
    // tedious to port to OCaml
}
```

# A Multi-Language Program in OCaml and C

## **OCaml** business logic

```
let main () =
  let r = ref 42 in
  hash_ref r; (*written in C*)
  print_int !r
```

## **C** business logic

```
void hash_ptr(int * x) {
    // Implemented in OpenSSL
    // tedious to port to OCaml
}
```

## **C** glue code

```
value caml_hash_ref(value r) {
    int x = Int_val(Field(r, 0));
    hash_ptr(&x);
    Store_field(r, 0, Val_int(x));
    return Val_unit;
}
```

# A Multi-Language Program in OCaml and C

## **OCaml** business logic

```
let main () =
  let r = ref 42 in
  hash_ref r; (*written in C*)
  print_int !r
```

## **C** business logic

```
void hash_ptr(int * x) {
    // Implemented in OpenSSL
    // tedious to port to OCaml
}
```

## **OCaml** glue code

```
external hash_ref
  : int ref -> unit
  = "caml_hash_ref"
```

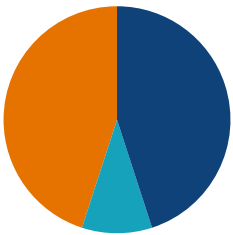
## **C** glue code

```
value caml_hash_ref(value r) {
    int x = Int_val(Field(r, 0));
    hash_ptr(&x);
    Store_field(r, 0, Val_int(x));
    return Val_unit;
}
```

# A Schematic Multi-Language Program

Most multi-language programs look like this:

**OCaml** business logic
oblivious of C



**C** business logic
oblivious of OCaml

**glue code**
where the languages actually interact

**We Need to Reason Language-Locally!**

# Our Contribution: Melocoton


**Common Approach:** program logic on top of semantics, **but**

- **Language Interaction:** new semantics and logic for glue code

# Our Contribution: Melocoton


**Common Approach:** program logic on top of semantics, **but**

- **Language Interaction:** new semantics and logic for glue code
- **Language Locality:** embed existing semantics and logics

*simplified/idealized versions of **OCaml** and **C**

# Our Contribution: Melocoton


**Common Approach:** program logic on top of semantics, **but**

- **Language Interaction:** new semantics and logic for glue code
- **Language Locality:** embed existing semantics and logics

*simplified/idealized versions of **OCaml** and **C**

# Language Interaction: Different Views of the Same Data

**OCaml** glue code

```
external hash_ref
  : int ref -> unit
  = "caml_hash_ref"
```

**C** glue code

```
value caml_hash_ref(value r) {
    int x = Int_val(Field(r, 0));
    hash_ptr(&x);
    Store_field(r, 0, Val_int(x));
    return Val_unit;
}
```

How is **OCaml** data accessed from **C** glue code?

# Language Interaction: Different Views of the Same Data

**OCaml** glue code

```
external hash_ref
  : int ref -> unit
  = "caml_hash_ref"
```

**C** glue code

```
value caml_hash_ref(value r) {
    int x = Int_val(Field(r, 0));
    hash_ptr(&x);
    Store_field(r, 0, Val_int(x));
    return Val_unit;
}
```

How is **OCaml** data accessed from **C** glue code?

High-level **OCaml** values are accessed..
..through a low-level **block** representation.

# Language Interaction: Semantics

High-level **OCaml** value $\sim_{\texttt{ML}}$ Low-level **block** representation

# Language Interaction: Semantics

High-level **OCaml** value $\sim_{\mathtt{ML}}$ Low-level **block** representation

integers $\sim_{\mathtt{ML}}$ integers

booleans $\sim_{\mathtt{ML}}$ integers (0 or 1)

*true* $\sim_{\mathtt{ML}}$ *1*

# Language Interaction: Semantics

High-level **OCaml** value $\sim_{\mathtt{ML}}$ Low-level **block** representation

integers $\sim_{\mathtt{ML}}$ integers

booleans $\sim_{\mathtt{ML}}$ integers (0 or 1)

arrays, refs $\sim_{\mathtt{ML}}$ blocks

$\mathit{true} \sim_{\mathtt{ML}} 1$

$\ell \sim_{\mathtt{ML}} \gamma$

# Language Interaction: Semantics

High-level **OCaml** value $\sim_{\mathtt{ML}}$ Low-level **block** representation

| | | |
|---|---|---|
| integers | $\sim_{\mathtt{ML}}$ | integers |
| booleans | $\sim_{\mathtt{ML}}$ | integers (0 or 1) |
| arrays, refs | $\sim_{\mathtt{ML}}$ | blocks |
| pairs | $\sim_{\mathtt{ML}}$ | blocks (of size 2) |

$\mathit{true} \sim_{\mathtt{ML}} 1$

$\ell \sim_{\mathtt{ML}} \gamma$

# Language Interaction: Semantics

| High-level **OCaml** value | $\sim_{\mathtt{ML}}$ | Low-level **block** representation |
|---|---|---|
| integers | $\sim_{\mathtt{ML}}$ | integers |
| booleans | $\sim_{\mathtt{ML}}$ | integers (0 or 1) |
| arrays, refs | $\sim_{\mathtt{ML}}$ | blocks |
| pairs | $\sim_{\mathtt{ML}}$ | blocks (of size 2) |
| lists | $\sim_{\mathtt{ML}}$ | block-based linked lists |

$\mathit{true} \sim_{\mathtt{ML}} 1$

$\ell \sim_{\mathtt{ML}} \gamma$

# Language Interaction: Semantics

High-level **OCaml** value $\sim_{\mathtt{ML}}$ Low-level **block** representation

| | | |
|---|---|---|
| integers | $\sim_{\mathtt{ML}}$ | integers |
| booleans | $\sim_{\mathtt{ML}}$ | integers (0 or 1) |
| arrays, refs | $\sim_{\mathtt{ML}}$ | blocks |
| pairs | $\sim_{\mathtt{ML}}$ | blocks (of size 2) |
| lists | $\sim_{\mathtt{ML}}$ | block-based linked lists |

$\mathit{true} \sim_{\mathtt{ML}} 1$

$\ell \sim_{\mathtt{ML}} \gamma$

$\lambda_{\mathtt{ML+C}}$ **Semantics**

$\sigma : \mathit{Heap}_{\mathtt{ML}}$

$\zeta : \mathit{BlockHeap}$

# Language Interaction: Semantics

High-level **OCaml** value $\sim_{\text{ML}}$ Low-level **block** representation

| | | |
|---|---|---|
| integers | $\sim_{\text{ML}}$ | integers |
| booleans | $\sim_{\text{ML}}$ | integers (0 or 1) |
| arrays, refs | $\sim_{\text{ML}}$ | blocks |
| pairs | $\sim_{\text{ML}}$ | blocks (of size 2) |
| lists | $\sim_{\text{ML}}$ | block-based linked lists |

$\mathit{true} \sim_{\text{ML}} 1$

$\ell \sim_{\text{ML}} \gamma$

$\lambda_{\text{ML+C}}$ **Semantics**

$\sigma : \mathit{Heap}_{\text{ML}}$ ⇄ $\zeta : \mathit{BlockHeap}$

switch at the language barrier

# Language Interaction: Semantics

| High-level **OCaml** value | $\sim_{\text{ML}}$ | Low-level **block** representation |
|---|---|---|
| integers | $\sim_{\text{ML}}$ | integers |
| booleans | $\sim_{\text{ML}}$ | integers (0 or 1) |
| arrays, refs | $\sim_{\text{ML}}$ | blocks |
| pairs | $\sim_{\text{ML}}$ | blocks (of size 2) |
| lists | $\sim_{\text{ML}}$ | block-based linked lists |

$\mathit{true} \sim_{\text{ML}} 1$

$\ell \sim_{\text{ML}} \gamma$

$\lambda_{\text{ML+C}}$ **Semantics**

$\sigma : \mathit{Heap}_{\text{ML}}$ ⇄ $\zeta : \mathit{BlockHeap}$

switch at the language barrier

# Language Interaction: Semantics

High-level **OCaml** value $\sim_{\text{ML}}$ Low-level **block** representation

| | | |
|---|---|---|
| integers | $\sim_{\text{ML}}$ | integers |
| booleans | $\sim_{\text{ML}}$ | integers (0 or 1) |
| arrays, refs | $\sim_{\text{ML}}$ | blocks |
| pairs | $\sim_{\text{ML}}$ | blocks (of size 2) |
| lists | $\sim_{\text{ML}}$ | block-based linked lists |

$\mathit{true} \sim_{\text{ML}} 1$

$\ell \sim_{\text{ML}} \gamma$

$\lambda_{\text{ML+C}}$ **Semantics**

$\sigma : \mathit{Heap}_{\text{ML}}$ ⇄ $\zeta : \mathit{BlockHeap}$

switch at the language barrier

# Language Interaction: Semantics

High-level **OCaml** value $\sim_{\text{ML}}$ Low-level **block** representation

| | | |
|---|---|---|
| integers | $\sim_{\text{ML}}$ | integers |
| booleans | $\sim_{\text{ML}}$ | integers (0 or 1) |
| arrays, refs | $\sim_{\text{ML}}$ | blocks |
| pairs | $\sim_{\text{ML}}$ | blocks (of size 2) |
| lists | $\sim_{\text{ML}}$ | block-based linked lists |

$\mathit{true} \sim_{\text{ML}} 1$

$\ell \sim_{\text{ML}} \gamma$

$\lambda_{\text{ML+C}}$ **Semantics**

$\sigma : \mathit{Heap}_{\text{ML}}$ $\quad$ $\zeta : \mathit{BlockHeap}$

switch at the language barrier

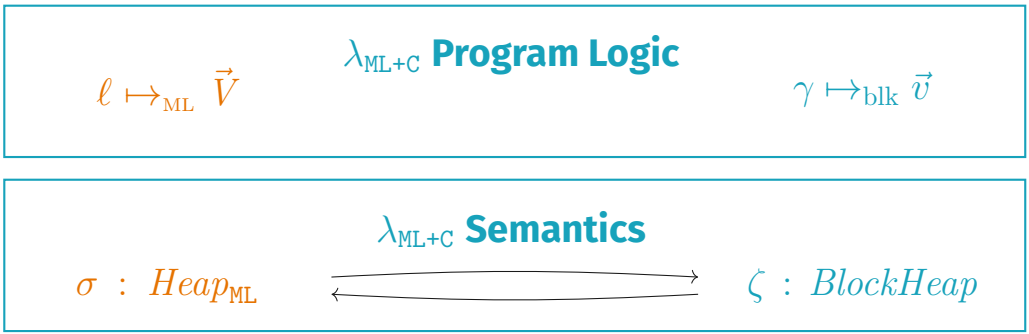
# Language Interaction: Program Logic, Take 1

# Language Interaction: Program Logic, Take 1

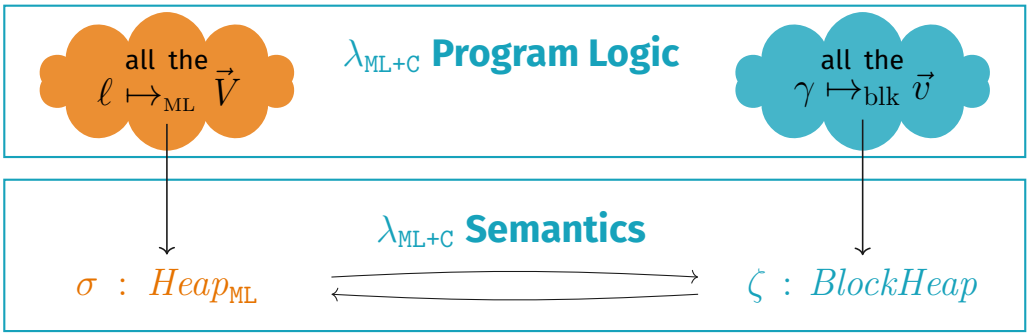
$\lambda_{\texttt{ML+C}}$ **Program Logic**

$\lambda_{\texttt{ML+C}}$ **Semantics**

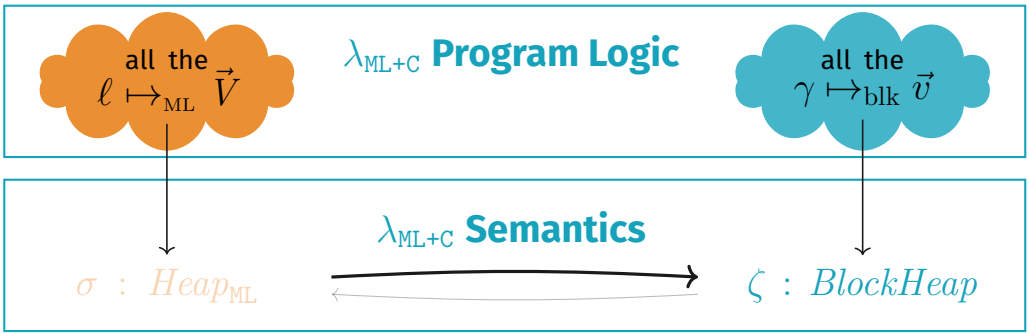
# Language Interaction: Program Logic, Take 1



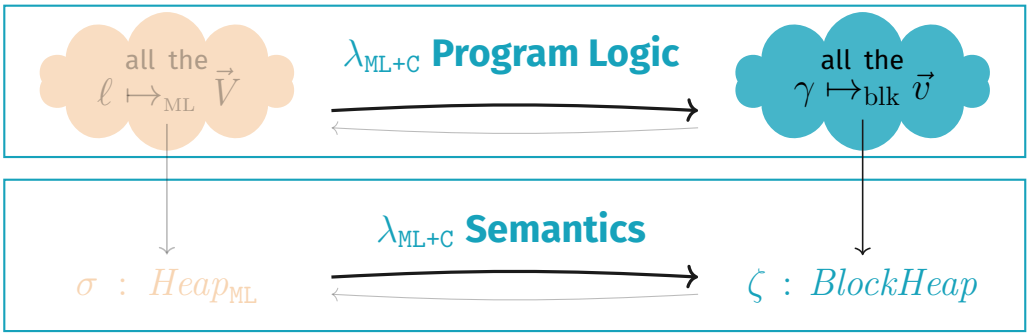
# Language Interaction: Program Logic, Take 1



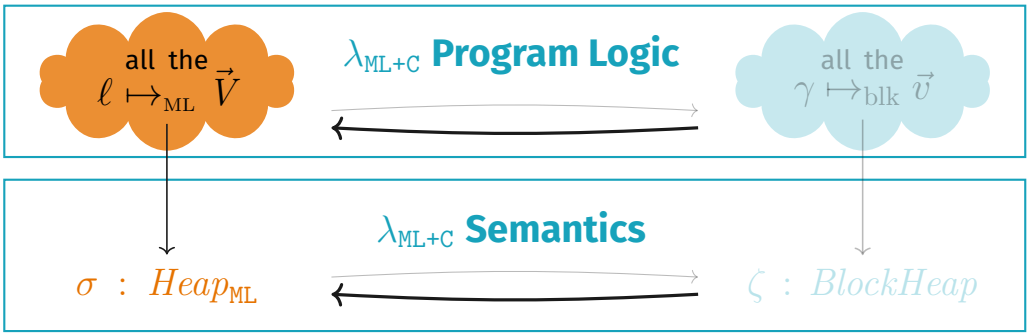
# Language Interaction: Program Logic, Take 1



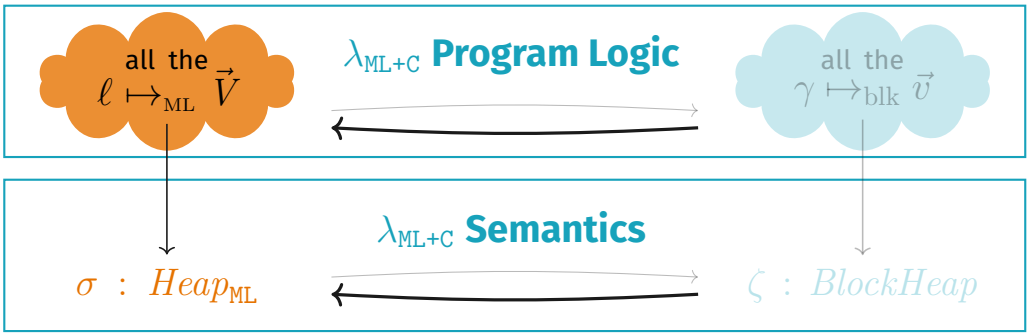
# Language Interaction: Program Logic, Take 1



# Language Interaction: Program Logic, Take 1



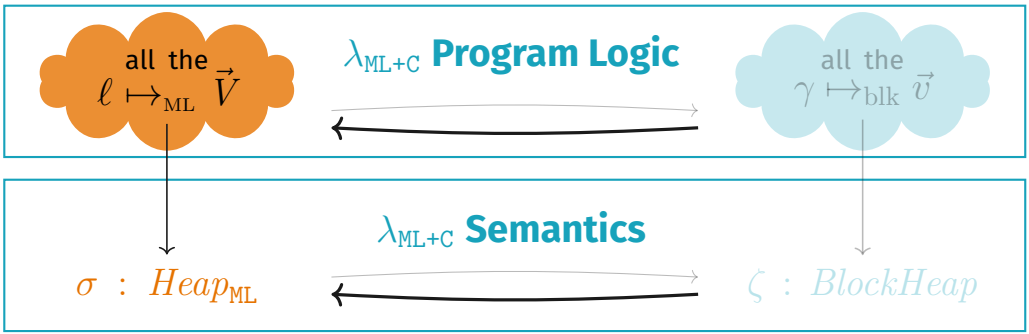
# Language Interaction: Program Logic, Take 1




ExtCall

$$\frac{\{\text{all}\}\ \mathbf{C}\ \text{function body}\ \{\text{all}\}}{\{\text{all}\}\ \text{call into}\ \mathbf{C}\ \{\text{all}\}}$$

# Language Interaction: Program Logic, Take 1



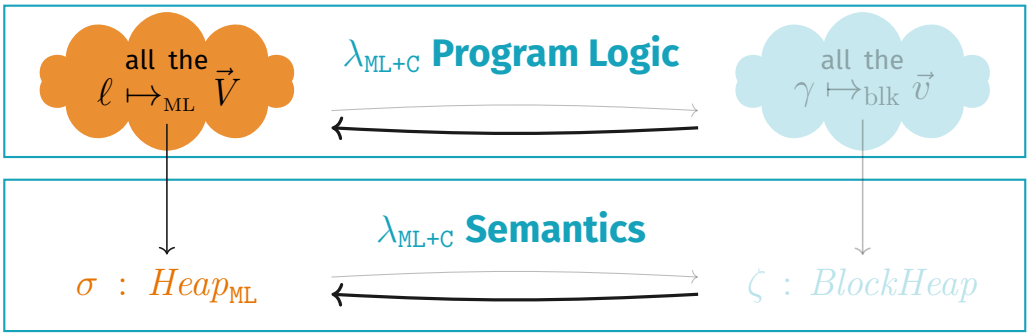

**ExtCall**

$$\frac{\{\text{all}\}\ \mathbf{C}\ \text{function body}\ \{\text{all}\}}{\{\text{all}\}\ \text{call into}\ \mathbf{C}\ \{\text{all}\}}$$

**Frame**

$$\frac{\{P\}\,e\,\{Q\}}{\{R * P\}\,e\,\{Q * R\}}$$

# Language Interaction: Program Logic, Take 1



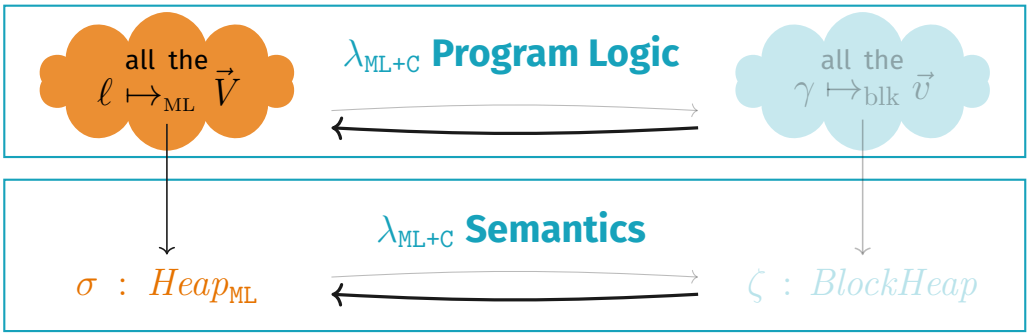

**ExtCall**

$$\frac{\{\text{all}\}\ \mathbf{C}\ \text{function body}\ \{\text{all}\}}{\{\text{all}\}\ \text{call into}\ \mathbf{C}\ \{\text{all}\}}$$

**Frame**

$$\frac{\{P\}\ \text{call into}\ \mathbf{C}\ \{Q\}}{\{R * P\}\ \text{call into}\ \mathbf{C}\ \{Q * R\}}$$

# Language Interaction: Program Logic, Take 1



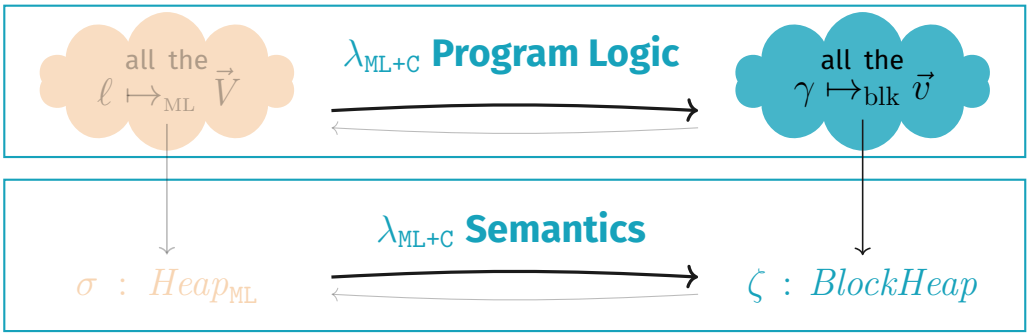

**ExtCall**

$$\frac{\{\text{all}\}\ \mathbf{C}\ \text{function body}\ \{\text{all}\}}{\{\text{all}\}\ \text{call into}\ \mathbf{C}\ \{\text{all}\}}$$

**Frame**

$$\frac{\{P\}\ \text{call into}\ \mathbf{C}\ \{Q\}}{\{\ell \mapsto_{\mathrm{ML}} \vec{V} * P\}\ \text{call into}\ \mathbf{C}\ \{Q * \ell \mapsto_{\mathrm{ML}} \vec{V}\}}$$

# Language Interaction: Program Logic, Take 1



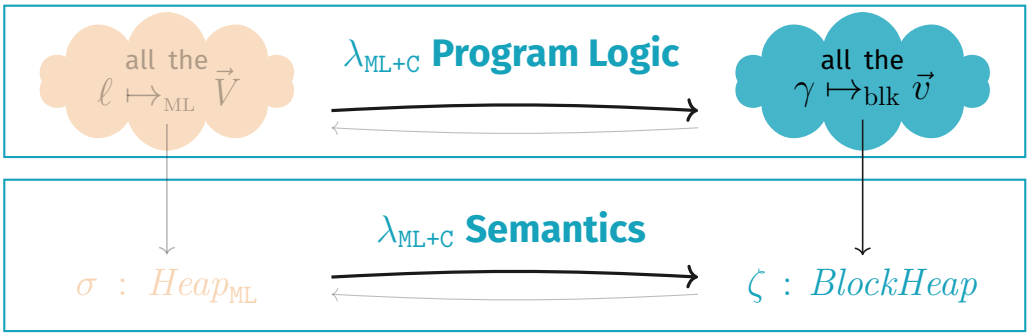

**ExtCall**

$$\frac{\{\text{all}\}\ \mathbf{C}\ \text{function body}\ \{\text{all}\}}{\{\text{all}\}\ \text{call into}\ \mathbf{C}\ \{\text{all}\}}$$

**Frame**

$$\frac{\{P\}\ \text{call into}\ \mathbf{C}\ \{Q\}}{\{\ell \mapsto_{\mathrm{ML}} \vec{V} * P\}\ \text{call into}\ \mathbf{C}\ \{Q * \ell \mapsto_{\mathrm{ML}} \vec{V}\}}$$

# Language Interaction: Program Logic, Take 1




ExtCall

{all} C function body {all}

{all} call into C {all}

Frame

$\{P\}$ call into C $\{Q\}$

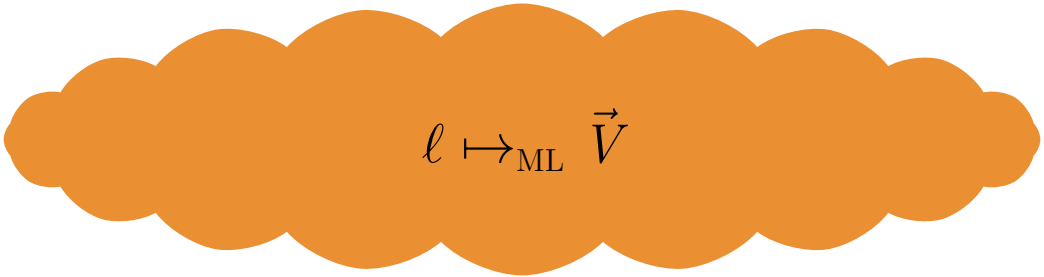
$\{\ell \mapsto_{\mathrm{ML}} \vec{V} * P\}$ call into C $\{Q * \ell \mapsto_{\mathrm{ML}} \vec{V}\}$

# Language Interaction: More Gradual Rules

**OCaml** points-tos *remain valid* when switching to **C**!

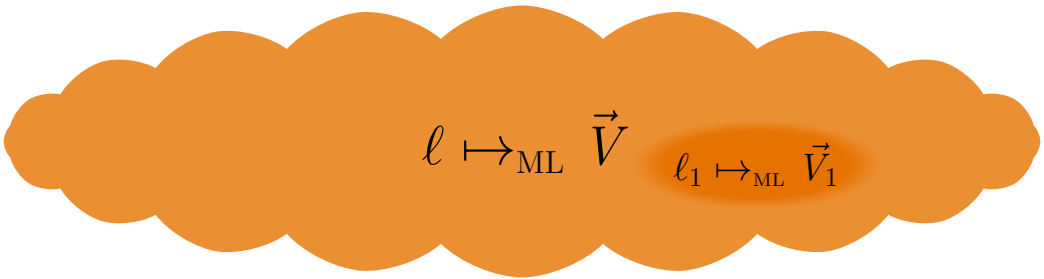
# Language Interaction: More Gradual Rules

**OCaml** points-tos *remain valid* when switching to **C**!



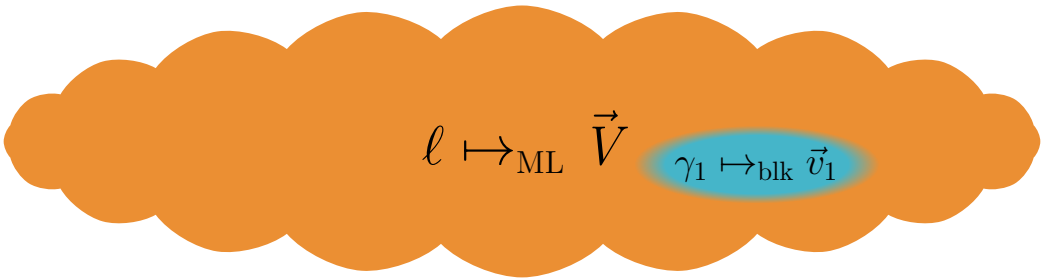
# Language Interaction: More Gradual Rules

**OCaml** points-tos *remain valid* when switching to **C**!



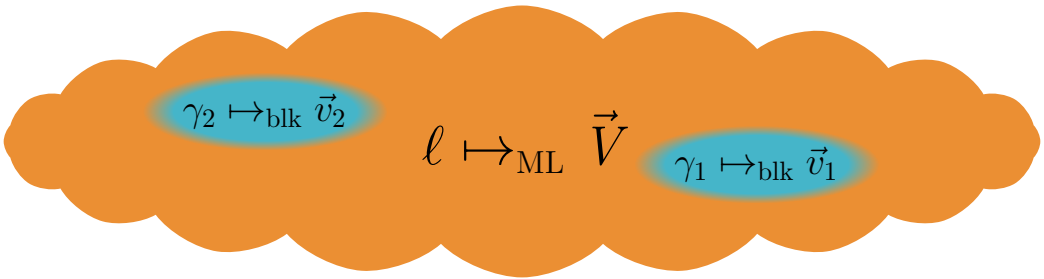
# Language Interaction: More Gradual Rules

**OCaml** points-tos *remain valid* when switching to **C**!



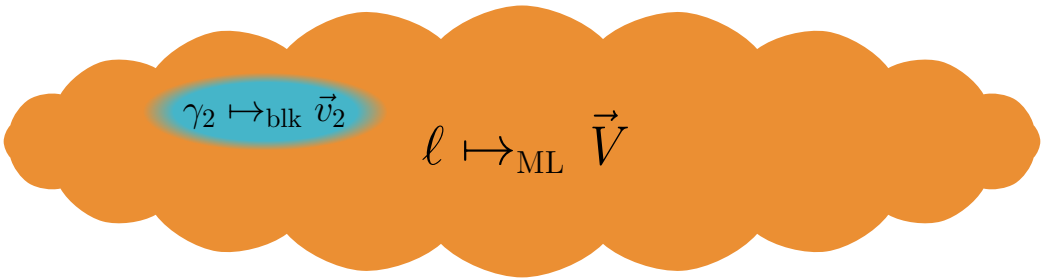
# Language Interaction: More Gradual Rules

**OCaml** points-tos *remain valid* when switching to **C**!



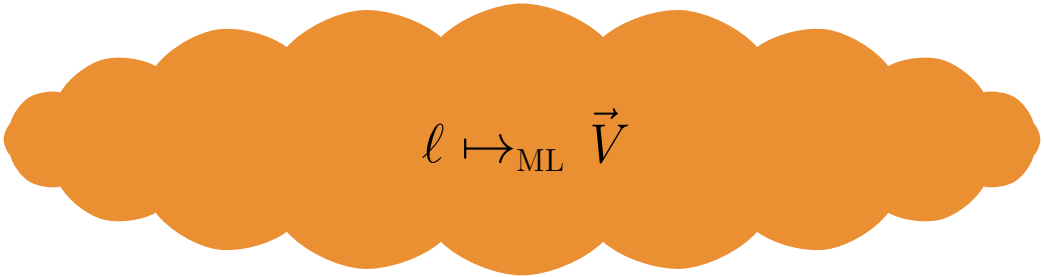
# Language Interaction: More Gradual Rules

**OCaml** points-tos *remain valid* when switching to **C**!



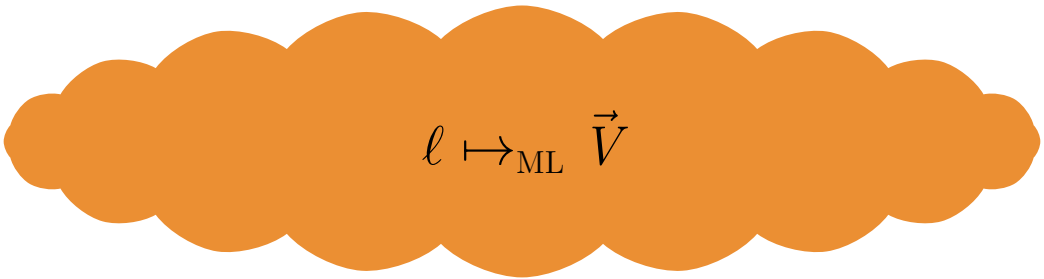
# Language Interaction: More Gradual Rules

**OCaml** points-tos *remain valid* when switching to **C**!



# Language Interaction: More Gradual Rules

**OCaml** points-tos *remain valid* when switching to **C**!




**View Reconciliation Rules for Converting On-Demand:**

$$\ell \mapsto_{\mathrm{ML}} \vec{V} \Rrightarrow \exists \gamma \vec{v}.\, \gamma \mapsto_{\mathrm{blk}} \vec{v} * \ell \sim_{\mathtt{ML}} \gamma * \vec{V} \sim_{\mathtt{ML}} \vec{v}$$

$$\vec{V} \sim_{\mathtt{ML}} \vec{v} * \gamma \mapsto_{\mathrm{blk}} \vec{v} \Rrightarrow \exists \ell\ .\, \ell \mapsto_{\mathrm{ML}} \vec{V} * \ell \sim_{\mathtt{ML}} \gamma$$

# Language Interaction: View Reconciliation

## View Reconciliation Rules

$$\ell \mapsto_{\mathrm{ML}} \vec{V} \;\; \text{==}\!\!\ast \;\; \exists \gamma \vec{v}.\, \gamma \mapsto_{\mathrm{blk}} \vec{v} * \ell \sim_{\mathtt{ML}} \gamma * \vec{V} \sim_{\mathtt{ML}} \vec{v}$$

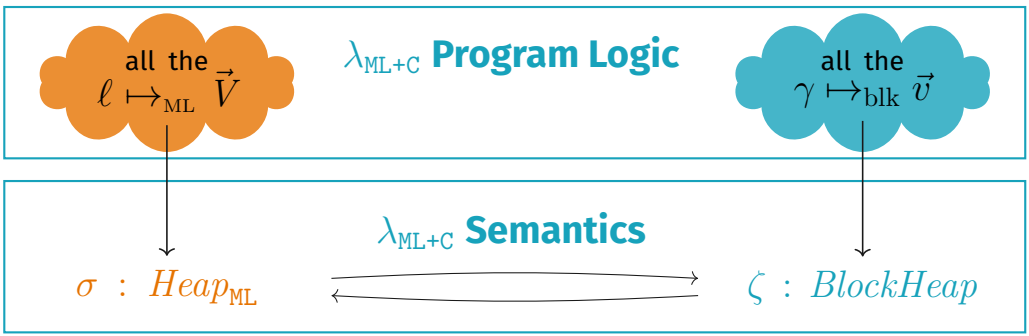
$$\vec{V} \sim_{\mathtt{ML}} \vec{v} * \gamma \mapsto_{\mathrm{blk}} \vec{v} \;\; \text{==}\!\!\ast \;\; \exists \ell \;\; .\, \ell \mapsto_{\mathrm{ML}} \vec{V} * \ell \sim_{\mathtt{ML}} \gamma$$

# Language Interaction: View Reconciliation

## View Reconciliation Rules

$$\ell \mapsto_{\mathtt{ML}} \vec{V} \;\; \text{-}\!\!\ast\!\!\ast \;\; \exists \gamma \vec{v}.\, \gamma \mapsto_{\mathrm{blk}} \vec{v} * \ell \sim_{\mathtt{ML}} \gamma * \vec{V} \sim_{\mathtt{ML}} \vec{v}$$

$$\vec{V} \sim_{\mathtt{ML}} \vec{v} * \gamma \mapsto_{\mathrm{blk}} \vec{v} \;\; \text{-}\!\!\ast\!\!\ast \;\; \exists \ell \;\; .\, \ell \mapsto_{\mathtt{ML}} \vec{V} * \ell \sim_{\mathtt{ML}} \gamma$$

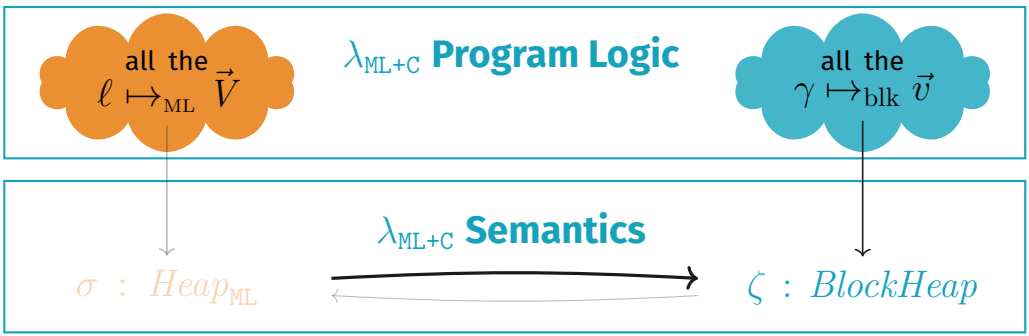


# Language Interaction: View Reconciliation

## View Reconciliation Rules

$$\ell \mapsto_{\mathtt{ML}} \vec{V} \;\; \text{⫲} \;\; \exists \gamma \vec{v}.\, \gamma \mapsto_{\mathrm{blk}} \vec{v} * \ell \sim_{\mathtt{ML}} \gamma * \vec{V} \sim_{\mathtt{ML}} \vec{v}$$

$$\vec{V} \sim_{\mathtt{ML}} \vec{v} * \gamma \mapsto_{\mathrm{blk}} \vec{v} \;\; \text{⫲} \;\; \exists \ell \;\; .\, \ell \mapsto_{\mathtt{ML}} \vec{V} * \ell \sim_{\mathtt{ML}} \gamma$$

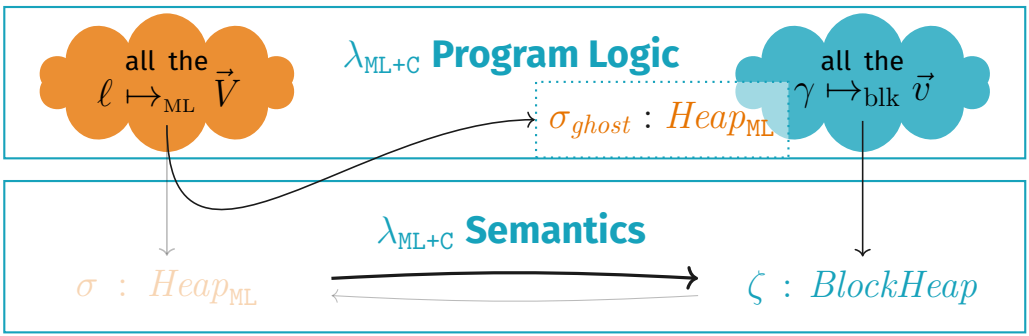


# Language Interaction: View Reconciliation

## View Reconciliation Rules

$$\ell \mapsto_{\mathtt{ML}} \vec{V} \;\; \text{-}\!\!\ast\!\!\ast \;\; \exists \gamma \vec{v}.\, \gamma \mapsto_{\mathtt{blk}} \vec{v} * \ell \sim_{\mathtt{ML}} \gamma * \vec{V} \sim_{\mathtt{ML}} \vec{v}$$

$$\vec{V} \sim_{\mathtt{ML}} \vec{v} * \gamma \mapsto_{\mathtt{blk}} \vec{v} \;\; \text{-}\!\!\ast\!\!\ast \;\; \exists \ell \;\; .\, \ell \mapsto_{\mathtt{ML}} \vec{V} * \ell \sim_{\mathtt{ML}} \gamma$$

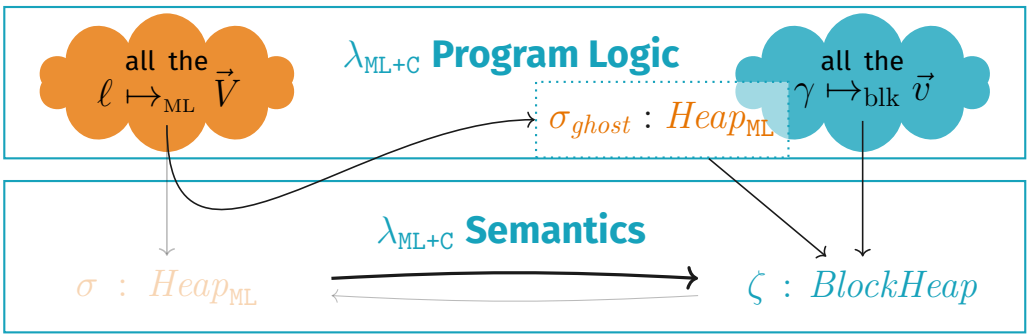


# Language Interaction: View Reconciliation

## View Reconciliation Rules

$$\ell \mapsto_{\mathtt{ML}} \vec{V} \;\; \text{-}\!\!\ast\!\!\ast \;\; \exists \gamma \vec{v}.\, \gamma \mapsto_{\mathtt{blk}} \vec{v} * \ell \sim_{\mathtt{ML}} \gamma * \vec{V} \sim_{\mathtt{ML}} \vec{v}$$

$$\vec{V} \sim_{\mathtt{ML}} \vec{v} * \gamma \mapsto_{\mathtt{blk}} \vec{v} \;\; \text{-}\!\!\ast\!\!\ast \;\; \exists \ell \;\; .\, \ell \mapsto_{\mathtt{ML}} \vec{V} * \ell \sim_{\mathtt{ML}} \gamma$$

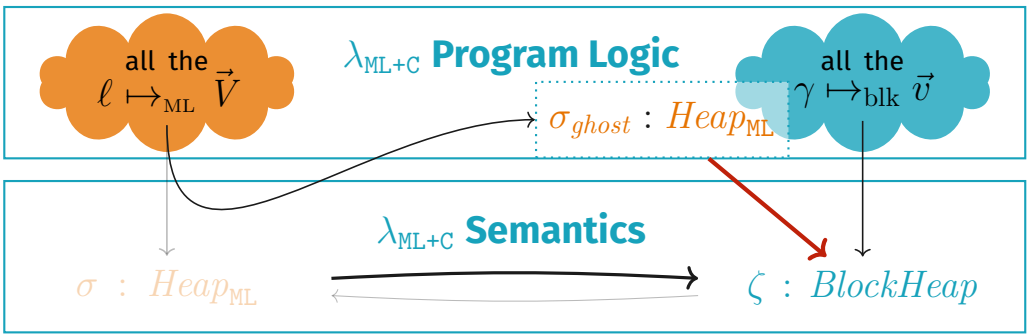


# Language Interaction: View Reconciliation

## View Reconciliation Rules

$$\ell \mapsto_{\mathtt{ML}} \vec{V} \;\; \text{-}\!\!\ast \;\; \exists \gamma \vec{v}.\, \gamma \mapsto_{\mathrm{blk}} \vec{v} * \ell \sim_{\mathtt{ML}} \gamma * \vec{V} \sim_{\mathtt{ML}} \vec{v}$$

$$\vec{V} \sim_{\mathtt{ML}} \vec{v} * \gamma \mapsto_{\mathrm{blk}} \vec{v} \;\; \text{-}\!\!\ast \;\; \exists \ell \;\; .\, \ell \mapsto_{\mathtt{ML}} \vec{V} * \ell \sim_{\mathtt{ML}} \gamma$$



# More in the paper …

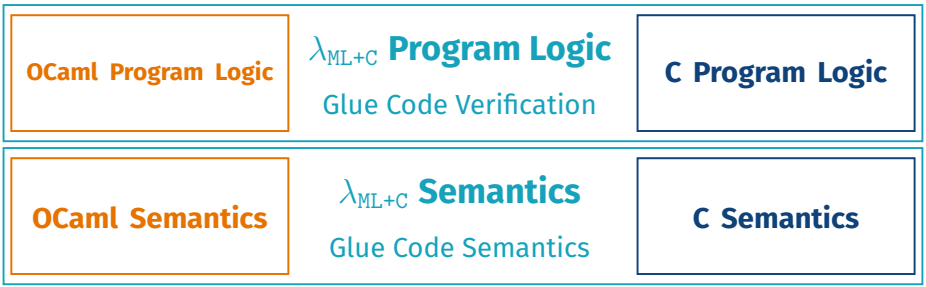
- Language-local reasoning for **external calls**.
- Additional **OCaml FFI features**: garbage collection, registering roots, custom blocks, callbacks, etc.
- **Case studies** utilising all of these features.
- **Step-indexed logical relation** to prove OCaml type safety of external C functions.

Transfinite

Iris

# Our Contribution: Melocoton

## Language Locality: Embed Existing Languages




## Language Interaction: View Reconciliation Rules

$$\ell \mapsto_{\text{ML}} \vec{V} \;\Rrightarrow\; \exists \gamma \vec{v}.\, \gamma \mapsto_{\text{blk}} \vec{v} * \ell \sim_{\text{ML}} \gamma * \vec{V} \sim_{\text{ML}} \vec{v}$$

$$\vec{V} \sim_{\text{ML}} \vec{v} * \gamma \mapsto_{\text{blk}} \vec{v} \;\Rrightarrow\; \exists \ell \;.\, \ell \mapsto_{\text{ML}} \vec{V} * \ell \sim_{\text{ML}} \gamma$$

https://melocoton-project.github.io