

# Verifying Tail Modulo Cons using Relational Separation Logic

Clément Allain  
Gabriel Scherer  
François Pottier

INRIA Paris

June 4, 2024

# Verifying Tail Modulo Cons using Relational Separation Logic

Program transformation implemented in the  
OCAML compiler by Frédéric Bour, Basile  
Clément & Gabriel Scherer.

## Verifying Tail Modulo Cons using Relational Separation Logic

Formalize the transformation and its soundness.

## Verifying Tail Modulo Cons using Relational Separation Logic

Prove soundness using an adequate IRIS binary logical relation à la SIMULIRIS.

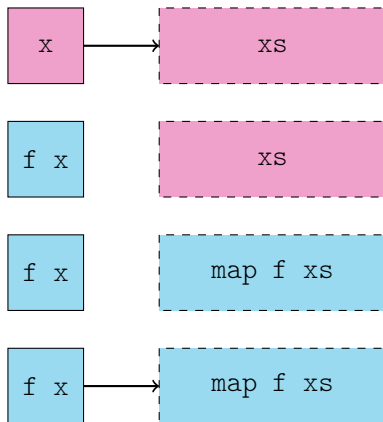
## The map problem: natural implementation

```
let rec map f xs =  
  match xs with  
  | [] →  
    []  
  | x :: xs →  
    let y = f x in  
    y :: map f xs
```

```
# List.init 250_000 (fun _ → ())  
|> map Fun.id  
|> ignore  
;;
```

Stack overflow during evaluation (looping recursion?).

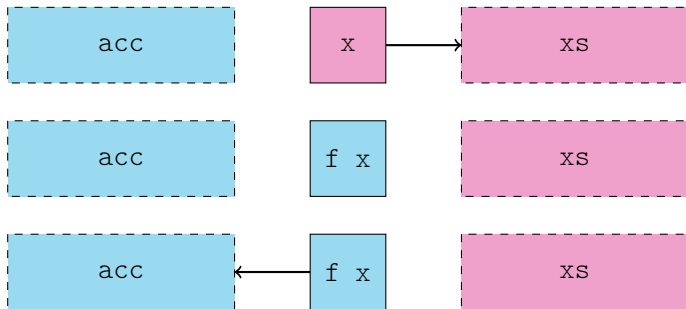
## The map problem: natural implementation



## The map problem: APS implementation

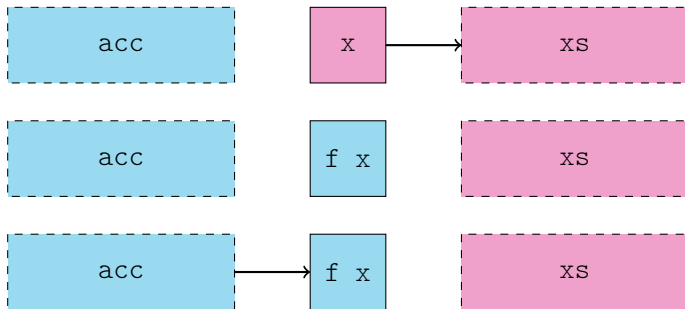
```
let rec map ys f xs =  
  match xs with  
  | [] →  
    List.rev ys  
  | x :: xs →  
    let y = f x in  
    map (y :: ys) f xs  
let map xs =  
  map [] f xs  
  
# List.init 250_000 (fun _ → ())  
|> map Fun.id  
|> ignore  
;;  
- : unit = ()
```

## The map problem: APS implementation





## The map problem: DPS implementation



## The map problem: DPS implementation

```
let rec map_dps dst f xs =      let map f xs =
  match xs with
  | [] →
    set_field dst 1 []
  | x :: xs →
    let y = f x in
    let dst' = y :: [] in
    set_field dst 1 dst' ;
    map_dps dst' f xs

# List.init 250_000 (fun _ → ())
|> map Fun.id
|> ignore
;;
- : unit = ()
```

## The map problem: TMC

```
let[@tail_mod_cons] rec map f xs =  
  match xs with  
  | [] →  
    []  
  | x :: xs →  
    let y = f x in  
    y :: map f xs
```

```
# List.init 250_000 (fun _ → ())  
|> map Fun.id  
|> ignore  
;;  
- : unit = ()
```

# DATA LANG: syntax

Index	$\ni$	$i$	$::=$	$0 \mid 1 \mid 2$
Tag	$\ni$	$t$		
$\mathbb{B}$	$\ni$	$b$		
$\mathbb{L}$	$\ni$	$\ell$		
$\mathbb{F}$	$\ni$	$f$		
$\mathbb{X}$	$\ni$	$x, y$		
Val	$\ni$	$v, w$	$::=$	$() \mid i \mid t \mid b \mid \ell \mid @f$
Expr	$\ni$	$e$	$::=$	$v \mid x \mid \text{let } x = e_1 \text{ in } e_2 \mid e_1 \overline{e_2}$ $\mid e_1 = e_2 \mid \text{if } e_0 \text{ then } e_1 \text{ else } e_2$ $\mid \{t, e_1, e_2\}$ $\mid e_1.(e_2) \mid e_1.(e_2) \leftarrow e_3$
Def	$\ni$	$d$	$::=$	$\text{rec } \overline{x} = e$
Prog	$\ni$	$p$	$::=$	$\mathbb{F} \xrightarrow{\text{fn}} \text{Def}$
State	$\ni$	$\sigma$	$::=$	$\mathbb{L} \xrightarrow{\text{fn}} \text{Val}$
Config	$\ni$	$\rho$	$::=$	$\text{Expr} \times \text{State}$

## DATA LANG: map

```
map := rec f xs =  
  match xs with  
  | [] →  
    []  
  | x :: xs →  
    let y = f x in  
    y :: @map f xs
```

# TMC transformation

$$e_s \xrightarrow[\text{dir}]{\xi} e_t$$

$$d_s \xrightarrow[\text{dir}]{\xi} d_t$$

$$(e_{dst}, e_{idx}, e_s) \xrightarrow[\text{dps}]{\xi} e_t$$

$$d_s \xrightarrow[\text{dps}]{\xi} d_t$$

$$p_s \rightsquigarrow p_t$$

## TMC transformation: map

```
map := rec f xs =  
  match xs with  
  | [] →  
    []  
  | x :: xs →  
    let y = f x in  
    let dst = y :: ■ in  
    @map_dps dst 2 f xs ;  
    dst
```

```
map_dps := rec dst idx f xs =  
  match xs with  
  | [] →  
    dst.(idx) ← []  
  | x :: xs →  
    let y = f x in  
    let dst' = y :: ■ in  
    dst.(idx) ← dst' ;  
    @map_dps dst' 2 f xs
```

## Transformation soundness

$p_s \rightsquigarrow p_t$       program  $p_s$  transforms into program  $p_t$



$p_s \sqsupseteq p_t$       program  $p_t$  refines program  $p_s$   
(termination-preserving refinement)



# Transformation soundness

$p_s \rightsquigarrow p_t$       program  $p_s$  transforms into program  $p_t$



$p_s \gtrsim p_t$       program  $p_t$  simulates program  $p_s$   
(*relational separation logic*, SIMULIRIS)



$p_s \sqsupseteq p_t$       program  $p_t$  refines program  $p_s$   
(termination-preserving refinement)

## Specification in separation logic

$$\frac{\{\text{???}\}}{\frac{\text{@map } v_s \gtrsim \text{@map } v_t}{\{\text{???}\}}}$$

$$\frac{\{\text{???}\}}{\frac{\text{@map } v_s \gtrsim \text{@map\_dps } \ell \ i \ v_t}{\{\text{???}\}}}$$

## Direct transformation

$$\frac{\frac{\{v_s \approx v_t\}}{\text{@map } v_s \gtrsim \text{@map } v_t}}{\{w_s, w_t. w_s \approx w_t\}}$$

RELDIR (SIMULIRIS)

$$f \in \text{dom}(p_s)$$

$$v_s \approx v_t$$

$$\frac{\forall w_s, w_t. w_s \approx w_t \rightarrow \Phi(w_s, w_t)}{\text{@f } v_s \gtrsim \text{@f } v_t [\Phi]}$$

# DPS transformation

$$\frac{\frac{\{v_s \approx v_t * (\ell + i) \mapsto_t \blacksquare\}}{\textcircled{\text{map}} \, v_s \gtrsim \textcircled{\text{map\_dps}} \, \ell \, i \, v_t}}{\{(), w_t. \exists w_t. w_s \approx w_t * (\ell + i) \mapsto_t w_t\}}$$

RELDPS

$$\frac{\begin{array}{c} \xi[f] = f_{dps} \\ \overline{v_s} \approx \overline{v_t} \\ \ell \mapsto_t \overline{v} \\ \forall w_s, w_t. w_s \approx w_t \multimap \ell \mapsto_t \overline{v}[i \mapsto w_t] \multimap \Phi(w_s, ()) \end{array}}{\textcircled{f} \, \overline{v_s} \gtrsim \textcircled{f_{dps}} \, \ell \, i \, \overline{v_t} \, [\Phi]}$$

RELPROTOCOL

$$\frac{\textcolor{red}{X}(e_s, e_t, \Psi) \quad \forall e'_s, e'_t. \Psi(e'_s, e'_t) \multimap e'_s \gtrsim e'_t \langle \textcolor{red}{X} \rangle \, [\Phi]}{e_s \gtrsim e_t \langle \textcolor{red}{X} \rangle \, [\Phi]}$$

# Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

---

$$@map\ f_s\ xs_s$$

$$\gtrsim$$

$$@map\ f_t\ xs_t$$

# Proof sketch

$$\begin{array}{c}
 f_s \approx f_t \qquad \qquad \qquad xs_s \approx xs_t \\
 \\
 \text{REL PURE} \\
 \frac{e_s \xrightarrow[\text{pure}]{p_s} e'_s \quad e_t \xrightarrow[\text{pure}]{p_t} e'_t \quad e'_s \gtrsim e'_t [\Phi]}{e_s \gtrsim e_t [\Phi]} \\
 \\
 @map \ f_s \ xs_s \qquad \qquad \qquad \gtrsim \qquad \qquad \qquad @map \ f_t \ xs_t
 \end{array}$$

# Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

---

```
match xs_s with
| [] →
  []
| x :: xs' →
  let y = f_s x in
  y :: @map f_s xs'
```

$$\gtrsim$$

```
match xs_t with
| [] →
  []
| x :: xs' →
  let y = f_t x in
  let dst = y :: ■ in
  @map_dps dst 2 f_t xs' ;
  dst
```

# Proof sketch

RELMATCH

$$e_{s0} \gtrsim e_{t0} \{ \approx \}$$

$$e_{s1} \gtrsim e_{t1} [\Phi]$$

$$\frac{\forall x_s, x_t, xs_s, xs_t. x_s \approx x_t \multimap xs_s \approx xs_t \multimap e_{s2} \gtrsim e_{t2} [\Phi]}{\text{match } e_{s0} \text{ with } \begin{array}{l} | [] \rightarrow e_{s1} \\ | x_s :: xs_s \rightarrow e_{s2} \end{array} \gtrsim \text{match } e_{t0} \text{ with } \begin{array}{l} | [] \rightarrow e_{t1} \\ | x_t :: xs_t \rightarrow e_{t2} \end{array} [\Phi]}$$

y :: @map f<sub>s</sub> xs'

@map\_dps dst 2 f<sub>t</sub> xs' ;  
dst



# Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

---

[ ]

$\wedge$

[ ]

## Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

---

```
let y = f_s x_s in  
y :: @map f_s xs'_s
```

$$\gtrsim$$

```
let y = f_t x_t in  
let dst = y :: ■ in  
@map_dps dst 2 f_t xs'_t ;  
dst
```

# Proof sketch

$$f_s \approx f_t \qquad x_{s_s} \approx x_{s_t}$$

$$x_s \approx x_t \qquad x_{s'_s} \approx x_{s'_t}$$

$$\frac{\text{RELVCALLSIMILAR} \quad f_s \approx f_t \quad x_s \approx x_t}{f_s \ x_s \gtrsim f_t \ x_t \ \{\approx\}}$$

```
let y = f_s x_s in
y :: @map f_s xs'_s
```

 $\gtrsim$ 

```
let dst = y :: ■ in
@map_dps dst 2 f_t xs'_t ;
dst
```

## Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

$$y_s \approx y_t$$

---

```
let y = y_s in  
y :: @map f_s xs'_s
```

$$\gtrsim$$

```
let y = y_t in  
let dst = y :: ■ in  
@map_dps dst 2 f_t xs'_t ;  
dst
```

## Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

$$y_s \approx y_t$$

---

$$y_s \quad :: \quad @map \quad f_s \quad xs'_s \quad \quad \quad \geq \quad \quad \quad \mathbf{let} \quad dst = y_t \quad :: \quad \blacksquare \quad \mathbf{in} \\ @map\_dps \quad dst \quad 2 \quad f_t \quad xs'_t \quad ; \\ dst$$

## Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

RELTGTCONS

$$\frac{\forall \ell. \ell \mapsto_t (\text{CONS}, v_1, v_2) \multimap e_s \gtrsim \ell [\Phi]}{e_s \gtrsim v_1 :: v_2 [\Phi]}$$

$y_s :: \text{@map } f_s \text{ } xs'_s$

$\gtrsim$

```
let dst = y_t :: ■ in
@map_dps dst 2 f_t xs'_t ;
dst
```

## Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

$$y_s \approx y_t$$

$$\ell_t \mapsto_t (\text{CONS}, y_t, \blacksquare)$$

---

$$y_s :: @map\ f_s\ xs'_s \quad \geq \quad \begin{array}{l} \text{let } dst = \ell_t \text{ in} \\ @map\_dps\ dst\ 2\ f_t\ xs'_t\ ; \\ dst \end{array}$$

## Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

RELTGTPURE

$$\frac{e_t \xrightarrow[pure]{p_t} e'_t \quad e_s \gtrsim e'_t [\Phi]}{e_s \gtrsim e_t [\Phi]}$$

$y_s :: @map \text{f}_s \text{ xs}'_s$

$\gtrsim$

```
let dst = l_t in
@map_dps dst 2 f_t xs'_t ;
dst
```



## Proof sketch

$$f_s \approx f_t \qquad xs_s \approx xs_t$$

$$x_s \approx x_t \qquad xs'_s \approx xs'_t$$

$$y_s \approx y_t$$

$$\ell_t \mapsto_t (\text{CONS}, y_t, \blacksquare)$$

---

$$y_s :: @\text{map} \ f_s \ xs'_s \quad \geq \quad @\text{map\_dps} \ \ell_t \ 2 \ f_t \ xs'_t \ ;$$

# Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

RELDPS2

$$\xi[f] = f_{dps}$$

$$\overline{v_s} \approx \overline{v_t}$$

$$\ell \mapsto_t (t, v_1, v_2)$$

$$\forall w_s, w_t. w_s \approx w_t \multimap \ell \mapsto_t (t, v_1, w_t) \multimap \Phi(w_s, ( ))$$

$$\textcircled{f} \overline{v_s} \gtrsim \textcircled{f_{dps}} \ell \textcircled{2} \overline{v_t} [\Phi]$$

$$y_s :: y_{s_s}$$

$$\gtrsim$$

$$\textcircled{\text{map\_dps}} \ell_t \textcircled{2} f_t xs'_t ;$$

$$\ell_t$$

## Proof sketch

$$f_s \approx f_t$$

$$xS_s \approx xS_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

$$y_s \approx y_t$$

$$ys_s \approx ys_t$$

$$\ell_t \mapsto_t (\text{CONS}, y_t, y_{s_t})$$

$$y_s \quad :: \quad y s_s$$

$\wedge 2$

$$(\quad) ; \ell_t$$

## Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

$$y_s \approx y_t$$

$$ys_s \approx ys_t$$

$$\ell_t \mapsto_t (\text{CONS}, y_t, ys_t)$$

---

$$y_s \quad :: \quad ys_s$$

$$\succeq$$

$$\ell_t$$

## Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

RELSRCCONS

$$\frac{\forall \ell. \ell \mapsto_s (\text{CONS}, v_1, v_2) \rightarrow * \ell \gtrsim e_t [\Phi]}{v_1 :: v_2 \gtrsim e_t [\Phi]}$$

$$v_1 :: v_2 \gtrsim e_t [\Phi]$$

---

 $\ell_s$  $\gtrsim$  $\ell_t$

## Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

$$y_s \approx y_t$$

$$ys_s \approx ys_t$$

$$\ell_t \mapsto_t (\text{CONS}, y_t, ys_t)$$

$$\ell_s \mapsto_s (\text{CONS}, y_s, ys_s)$$

---

 $\ell_s$  $\geq$  $\ell_t$

# Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

RELBIIINSERT

$$\ell_s \mapsto_s \overline{v_s}$$

$$\ell_t \mapsto_t \overline{v_t}$$

$$\overline{v_s} \approx \overline{v_t}$$

$$\ell_s \approx \ell_t \multimap e_s \gtrsim e_t [\Phi]$$

$$\hline e_s \gtrsim e_t [\Phi]$$

---

 $\ell_s$  $\gtrsim$  $\ell_t$

# Proof sketch

$$f_s \approx f_t$$

$$xs_s \approx xs_t$$

$$x_s \approx x_t$$

$$xs'_s \approx xs'_t$$

$$y_s \approx y_t$$

$$ys_s \approx ys_t$$

$$\ell_s \approx \ell_t$$

---

 $\ell_s$  $\succeq$  $\ell_t$



## Concluding remarks

- ▶ The real proof deals with the *abstract* relational transformation.
- ▶ Details regarding the *undetermined evaluation order* of constructors were eluded.
- ▶ Other program transformations verified using *protocols*: APS, inlining.

Thank you for your attention!