Verifying Tail Modulo Cons using Relational Separation Logic

Clément Allain
Gabriel Scherer
François Pottier

INRIA Paris

June 5, 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

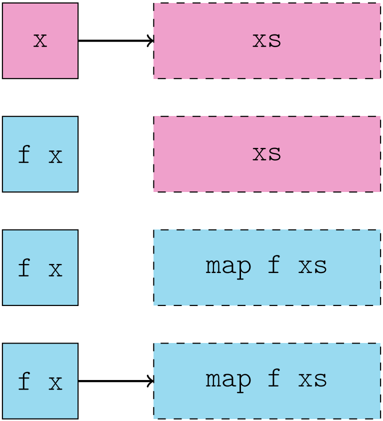
Prove soundness using an adequate IRIS binary logical relation à la SIMULIRIS.

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

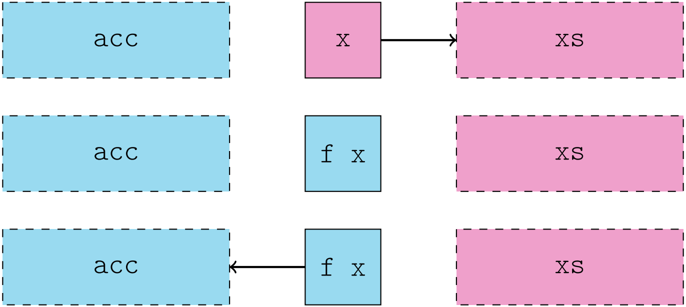


The map problem: APS implementation

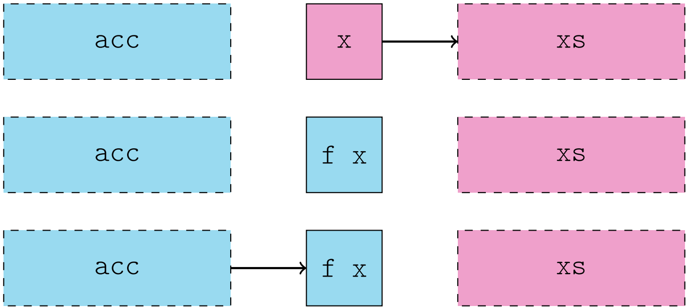
```
let rec map ys f xs =
  match xs with
  | [] →
      List.rev ys
  | x :: xs →
      let y = f x in
      map (y :: ys) f xs
let map xs =
  map [] f xs
```

```
# 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 =
  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
```

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

```
# 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 = ()
```

DataLang: syntax

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

DataLang: map

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

TMC transformation

$$e_s \overset{\xi}{\underset{\text{dir}}{\rightsquigarrow}} e_t \qquad\qquad d_s \overset{\xi}{\underset{\text{dir}}{\rightsquigarrow}} d_t$$

$$(e_{dst}, e_{idx}, e_s) \overset{\xi}{\underset{\text{dps}}{\rightsquigarrow}} e_t \qquad\qquad d_s \overset{\xi}{\underset{\text{dps}}{\rightsquigarrow}} d_t$$

$$p_s \rightsquigarrow p_t$$

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

$\Downarrow$

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

Transformation soundness

$p_s \rightsquigarrow p_t$	program p_s transforms into program p_t
$\Downarrow$	
$p_s \gtrsim p_t$	program p_t simulates program p_s (*relational separation logic*, Simuliris)
$\Downarrow$	
$p_s \sqsupseteq p_t$	program p_t refines program p_t (termination-preserving refinement)

Specification in separation logic

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

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

Direct transformation

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

RelDir (Simuliris)

$$\frac{f \in \mathrm{dom}(p_s) \qquad v_s \approx v_t \qquad \forall\, w_s, w_t.\, w_s \approx w_t \mathrel{-\!\!*} \Phi(w_s, w_t)}{\texttt{@}f\ v_s \gtrsim \texttt{@}f\ v_t\ [\Phi]}$$

DPS transformation

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

RelDPS

$$\frac{\xi[f] = f_{dps} \qquad \overline{v_s} \approx \overline{v_t} \qquad \ell \mapsto_t \overline{v} \qquad \forall\, w_s, w_t.\, w_s \approx w_t \mathrel{-\!\!*} \ell \mapsto_t \overline{v}[i \mapsto w_t] \mathrel{-\!\!*} \Phi(w_s, ())}{\texttt{@}f\ \overline{v_s} \gtrsim \texttt{@}f_{dps}\ \ell\ i\ \overline{v_t}\ [\Phi]}$$

RelProtocol

$$\frac{\mathrm{X}(e_s, e_t, \Psi) \qquad \forall\, e_s', e_t'.\, \Psi(e_s', e_t') \mathrel{-\!\!*} e_s' \gtrsim e_t' \, \langle \mathrm{X} \rangle\ [\Phi]}{e_s \gtrsim e_t\, \langle \mathrm{X} \rangle\ [\Phi]}$$

Proof sketch

$$\frac{f_s \approx f_t \qquad\qquad xs_s \approx xs_t}{\texttt{@map}\ f_s\ \ xs_s \quad \gtrsim \quad \texttt{@map}\ f_t\ \ xs_t}$$

Proof sketch

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

RelPure

$$\frac{e_s \xrightarrow[\text{pure}]{p_s} e'_s \qquad e_t \xrightarrow[\text{pure}]{p_t} e'_t \qquad e'_s \gtrsim e'_t\ [\Phi]}{e_s \gtrsim e_t\ [\Phi]}$$

`@map` f_s xs_s $\qquad \gtrsim \qquad$ `@map` f_t xs_t

Proof sketch

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

```
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
```

RelMatch

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

```
y :: @map f_s xs'                 @map_dps dst 2 f_t xs' ;
                                  dst
```

Proof sketch

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

`[]` $\gtrsim$ `[]`

Proof sketch

$$\dfrac{\begin{array}{cc} f_s \approx f_t & xs_s \approx xs_t \\[1em] x_s \approx x_t & xs'_s \approx xs'_t \end{array}}{\begin{array}{l}\textbf{let}\ \texttt{y}\ \texttt{=}\ f_s\ x_s\ \textbf{in} \\ \texttt{y :: @map}\ f_s\ xs'_s\end{array} \quad \gtrsim \quad \begin{array}{l}\textbf{let}\ \texttt{y}\ \texttt{=}\ f_t\ x_t\ \textbf{in} \\ \textbf{let}\ \texttt{dst = y ::}\ \blacksquare\ \textbf{in} \\ \texttt{@map\_dps dst 2}\ f_t\ xs'_t\ \texttt{;} \\ \texttt{dst}\end{array}}$$

Proof sketch

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

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

RelvCallSimilar

$$\frac{f_s \approx f_t \qquad x_s \approx x_t}{f_s\ x_s \gtrsim f_t\ x_t\ \{\approx\}}$$

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

$\gtrsim$

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

Proof sketch

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

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

$$y_s \approx y_t$$

$$\begin{array}{l}\textbf{let}\ \texttt{y} = y_s\ \textbf{in}\\ \texttt{y :: @map}\ f_s\ xs'_s\end{array} \quad \gtrsim \quad \begin{array}{l}\textbf{let}\ \texttt{y} = y_t\ \textbf{in}\\ \textbf{let}\ \texttt{dst = y ::}\ \blacksquare\ \textbf{in}\\ \texttt{@map\_dps dst 2}\ f_t\ xs'_t\ \texttt{;}\\ \texttt{dst}\end{array}$$

Proof sketch

$$\frac{\begin{array}{cc} f_s \approx f_t & xs_s \approx xs_t \\ x_s \approx x_t & xs'_s \approx xs'_t \\ \multicolumn{2}{c}{y_s \approx y_t} \end{array}}{y_s \ \texttt{::} \ \texttt{@map} \ f_s \ xs'_s \quad \gtrsim \quad \begin{array}{l} \textbf{\texttt{let}} \ \texttt{dst} \ \texttt{=} \ y_t \ \texttt{::} \ \blacksquare \ \textbf{\texttt{in}} \\ \texttt{@map\_dps} \ \texttt{dst} \ \texttt{2} \ f_t \ xs'_t \ \texttt{;} \\ \texttt{dst} \end{array}}$$

Proof sketch

$f_s \approx f_t$ $\quad xs_s \approx xs_t$

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

$$\dfrac{\forall \ell.\, \ell \mapsto_t (\mathrm{CONS}, v_1, v_2) \mathrel{-\!\!*} e_s \gtrsim \ell\ [\Phi]}{e_s \gtrsim v_1 :: v_2\ [\Phi]}\ \textsc{RelTgtCons}$$

y_s `::` `@map` f_s xs'_s $\quad\gtrsim\quad$

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

Proof sketch

$$
\frac{f_s \approx f_t \qquad xs_s \approx xs_t \qquad x_s \approx x_t \qquad xs'_s \approx xs'_t \qquad y_s \approx y_t \qquad \ell_t \mapsto_t (\mathrm{CONS}, y_t, \blacksquare)}{y_s \; \texttt{::} \; \texttt{@map} \; f_s \; xs'_s \quad \gtrsim \quad \begin{array}{l}\textbf{\texttt{let}} \; \texttt{dst} = \ell_t \; \textbf{\texttt{in}} \\ \texttt{@map\_dps dst 2} \; f_t \; xs'_t \; \texttt{;} \\ \texttt{dst}\end{array}}
$$

Proof sketch

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

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

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

y_s `::` `@map` f_s xs'_s $\gtrsim$

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

Proof sketch

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

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

$$y_s \approx y_t$$

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

---

y_s `::` `@map` f_s xs'_s $\quad\gtrsim\quad$ `@map_dps` ℓ_t `2` f_t xs'_t `;` ℓ_t

Proof sketch

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

RelDPS2

$$\frac{\xi[f] = f_{dps} \qquad \overline{v_s} \approx \overline{v_t} \qquad \ell \mapsto_t (t, v_1, v_2) \qquad \forall\, w_s, w_t.\ w_s \approx w_t \twoheadrightarrow \ell \mapsto_t (t, v_1, w_t) \twoheadrightarrow \Phi(w_s, ())}{\texttt{@}f\ \overline{v_s} \gtrsim \texttt{@}f_{dps}\ \ell\ 2\ \overline{v_t}\ [\Phi]}$$

$y_s\ ::\ ys_s$ $\quad\gtrsim\quad$ $\texttt{@map\_dps}\ \ell_t\ 2\ f_t\ xs'_t\ ;\ \ell_t$

Proof sketch

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

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

$$y_s \approx y_t$$

$$ys_s \approx ys_t$$

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

$$y_s \;::\; ys_s \qquad \gtrsim \qquad (\,) \;;\; \ell_t$$

Proof sketch

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

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

$$y_s \approx y_t$$

$$ys_s \approx ys_t$$

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

$$y_s \;::\; ys_s \qquad \gtrsim \qquad \ell_t$$

Proof sketch

$f_s \approx f_t$ $\quad$ $xs_s \approx xs_t$

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

$$\frac{\forall \ell.\, \ell \mapsto_s (\mathrm{CONS}, v_1, v_2) \twoheadrightarrow \ell \gtrsim e_t\ [\Phi]}{v_1 :: v_2 \gtrsim e_t\ [\Phi]}\ \textsc{RelSrcCons}$$

ℓ_s $\quad \gtrsim \quad$ ℓ_t

Proof sketch

$$\frac{\begin{array}{cc} f_s \approx f_t & xs_s \approx xs_t \\ x_s \approx x_t & xs'_s \approx xs'_t \\ \multicolumn{2}{c}{y_s \approx y_t} \\ \multicolumn{2}{c}{ys_s \approx ys_t} \\ \multicolumn{2}{c}{\ell_t \mapsto_t (\mathrm{CONS}, y_t, ys_t)} \\ \multicolumn{2}{c}{\ell_s \mapsto_s (\mathrm{CONS}, y_s, ys_s)} \end{array}}{\ell_s \quad \gtrsim \quad \ell_t}$$

Proof sketch

$f_s \approx f_t$ $\quad$ $xs_s \approx xs_t$

$$\frac{\textsc{RelBijInsert} \quad \ell_s \mapsto_s \overline{v_s} \quad \ell_t \mapsto_t \overline{v_t} \quad \overline{v_s} \approx \overline{v_t} \quad \ell_s \approx \ell_t \twoheadrightarrow e_s \gtrsim e_t \;[\Phi]}{e_s \gtrsim e_t \;[\Phi]}$$

ℓ_s $\quad$ $\gtrsim$ $\quad$ ℓ_t

Proof sketch

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

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

$$y_s \approx y_t$$

$$ys_s \approx ys_t$$

$$\ell_s \approx \ell_t$$

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

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!

Simulation

$$
\mathrm{sim\text{-}body}_{\mathrm{X}} \coloneqq \begin{array}{l}
\lambda\, \mathit{sim}.\, \lambda\, \mathit{sim\text{-}inner}.\, \lambda\,(\Phi, e_s, e_t).\, \forall\, \sigma_s, \sigma_s.\, \mathrm{I}(\sigma_s, \sigma_t) \mathrel{-\!\!*} \Rrightarrow \\
\bigvee \left[\begin{array}{ll}
① & \mathrm{I}(\sigma_s, \sigma_t) * \Phi(e_s, e_t) \\
② & \mathrm{I}(\sigma_s, \sigma_t) * \text{strongly-stuck}_{p_s}(e_s) * \text{strongly-stuck}_{p_t}(e_s) \\
③ & \exists\, e_s', \sigma_s'.\, (e_s, \sigma_s) \xrightarrow{p_s}{}^{+} (e_s', \sigma_s') * \mathrm{I}(\sigma_s', \sigma_t) * \mathit{sim\text{-}inner}(\Phi, e_s', e_t) \\
④ & \text{reducible}_{p_t}(e_t, \sigma_t) * \forall\, e_t', \sigma_t'.\, (e_t, \sigma_t) \xrightarrow{p_t} (e_t', \sigma_t') \mathrel{-\!\!*} \Rrightarrow \\
 & \bigvee \left[\begin{array}{ll}
Ⓐ & \mathrm{I}(\sigma_s, \sigma_t') * \mathit{sim\text{-}inner}(\Phi, e_s, e_t') \\
Ⓑ & \exists\, e_s', \sigma_s'.\, (e_s, \sigma_s) \xrightarrow{p_s}{}^{+} (e_s', \sigma_s') * \\
 & \mathrm{I}(\sigma_s', \sigma_t') * \mathit{sim}(\Phi, e_s', e_t')
\end{array}\right. \\
⑤ & \exists\, K_s, e_s', K_t, e_t', \Psi. \\
 & e_s = K_s[e_s'] * e_t = K_t[e_t'] * \mathrm{X}(\Psi, e_s', e_t') * \mathrm{I}(\sigma_s, \sigma_t) * \\
 & \forall\, e_s'', e_t''.\, \Psi(e_s'', e_t'') \mathrel{-\!\!*} \mathit{sim\text{-}inner}(\Phi, K_s[e_s''], K_t[e_t''])
\end{array}\right.
\end{array}
$$

$$\mathrm{sim\text{-}inner}_{\mathrm{X}} \coloneqq \lambda\, \mathit{sim}.\, \mu\, \mathit{sim\text{-}inner}.\, \mathrm{sim\text{-}body}_{\mathrm{X}}(\mathit{sim}, \mathit{sim\text{-}inner})$$

$$\mathrm{sim}_{\mathrm{X}} \coloneqq \nu\, \mathit{sim}.\, \mathrm{sim\text{-}inner}_{\mathrm{X}}(\mathit{sim})$$

$$e_s \gtrsim e_t \,\langle \mathrm{X} \rangle\, [\Phi] \coloneqq \mathrm{sim}_{\mathrm{X}}(\Phi, e_s, e_t)$$

$$e_s \gtrsim e_t \,\langle \mathrm{X} \rangle\, \{\Phi\} \coloneqq e_s \gtrsim e_t \,\langle \mathrm{X} \rangle\, \left[\lambda(e_s', e_t').\, \exists\, v_s, v_t.\, e_s' = v_s * e_t' = v_t * \Phi(v_s, v_t)\right]$$

TMC protocol

$$
\begin{aligned}
\mathrm{X}_{\mathrm{dir}}(\Psi, e_s, e_t) \quad &\coloneqq \quad \exists\, f, v_s, v_t. \\
&\qquad f \in \mathrm{dom}(p_s) * \\
&\qquad e_s = \texttt{@}f\ \ v_s * e_t = \texttt{@}f\ \ v_t * v_s \approx v_t * \\
&\qquad \forall\, v_s', v_t'.\, v_s' \approx v_t' \mathrel{-\!\!*} \Psi(v_s', v_t')
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{X}_{\mathrm{DPS}}(\Psi, e_s, e_t) \quad &\coloneqq \quad \exists\, f, f_{dps}, v_s, \ell, i, v_t. \\
&\qquad f \in \mathrm{dom}(p_s) * \xi[f] = f_{dps} * \\
&\qquad e_s = \texttt{@}f\ \ v_s * e_t = \texttt{@}f_{dps}\ \ (\,(\ell\,,\, i)\,,\, v_t) * v_s \approx v_t * \\
&\qquad (\ell + i) \mapsto \blacksquare * \\
&\qquad \forall\, v_s', v_t'.\, (\ell + i) \mapsto v_t' * v_s' \approx v_t' \mathrel{-\!\!*} \Psi(v_s', (\,))
\end{aligned}
$$

$$
\mathrm{X}_{\mathrm{TMC}} \quad \coloneqq \quad \mathrm{X}_{\mathrm{dir}} \sqcup \mathrm{X}_{\mathrm{DPS}}
$$