The Next 700 Relational Program Logics

Kenji Maillard,
Cătălin Hriţcu, Exequiel Rivas, Antoine Van Muylder

Inria Paris

Paper conditionally accepted at POPL'20:
https://arxiv.org/abs/1907.05244

two (different) computational monads

relational monad morphism

relational specification monad

$$M_1, M_2 \xrightarrow{\theta_{rel}} W_{rel}$$

$M_1 A = S_1 \to A \times S_1$

$M_2 A = S_2 \to A \times S_2$

$W_{rel} A_1 A_2 = ((A_1 \times S_1) \times (A_2 \times S_2) \to P) \to S_1 \times S_2 \to P$

$\theta_{rel}(c_1, c_2) = \lambda post\ s_1\ s_2.\ post\ (c_1\ s_1,\ c_2\ s_2)$

exploit **syntactic similarity** between c_1 and c_2

c_1 and c_2 run **independently**, not something SMT solvable

Solution: define relational program logics,

using θ_{rel} for the semantics: $\vDash c_1 \sim c_2\ \{w\} = \theta_{rel}(c_1, c_2) \le w$

Rules defined using general recipe, $\forall M_1, M_2, \theta_{rel}, W_{rel}$

General recipe, 3 kinds of rules:

1. Rules from ambient dependent type theory
2. Rules for monadic constructs (sound for all)

$$\textsc{Ret}\ \frac{a_1 : A_1 \qquad a_2 : A_2}{\vdash \mathtt{ret}^{M_1}\, a_1 \sim \mathtt{ret}^{M_2}\, a_2 \ \left\{ \mathtt{ret}^{W} (a_1, a_2) \right\}} \qquad \textsc{Weaken}\ \frac{\vdash c_1 \sim c_2 \ \{ w \} \qquad w \leq w'}{\vdash c_1 \sim c_2 \ \{ w' \}}$$

$$\textsc{Bind}\ \frac{\vdash m_1 \sim m_2 \ \{ w^m \} \qquad \forall a_1, a_2 \vdash f_1\, a_1 \sim f_2\, a_2 \ \left\{ w^f\, (a_1, a_2) \right\}}{\vdash \mathtt{bind}^{M_1}\, m_1\, f_1 \sim \mathtt{bind}^{M_2}\, m_2\, f_2 \ \left\{ \mathtt{bind}^{W_{\mathrm{rel}}}\, w^m\, w^f \right\}}$$

3. Rules for effect-specific actions

$$\frac{}{\vdash \mathtt{get}\,() \sim \mathtt{ret}\, a_2 \ \{ \lambda\varphi\, (s_1, s_2).\ \varphi\, ((s_1, s_1), (a_2, s_2)) \}} \qquad \frac{}{\vdash \mathtt{put}\, s \sim \mathtt{ret}\, a_2 \ \{ \lambda\varphi\, (s_1, s_2).\ \varphi\, (((), s), (a_2, s_2)) \}}$$

Recipe for algebraic operations (soundness guaranteed):
unfold get and ret then apply θ_{rel} to them to obtain w

This works: state, nondet, IO, RHL (state+loops), RHTT

1st extension (work in progress)

needed for **probabilities**, nondet refinement, ...

$$\mathrm{W}_{\mathrm{rel}} A_1\, A_2 = ((A_1 \times A_2) \to [0,1]) \to [0,1]$$

$$\frac{p, q : [0,1] \qquad r \sim (\mathcal{B}_p, \mathcal{B}_q)}{\vdash \texttt{flip}\, p \sim \texttt{flip}\, q \left\{ \lambda post.\ \sum_{b_1, b_2} r(b_1, b_2) \cdot post(b_1, b_2) \right\}}$$

$$\theta_{\mathrm{rel}}(d_1, d_2) = \lambda post.\ \inf_{r \sim (d_1, d_2)} \sum_{b_1, b_2} r(b_1, b_2) \cdot post(b_1, b_2)$$

***Lax* relational monad morphism:**

$$\theta_{rel}\,(\texttt{bind}^{\mathrm{M}_1}\ m_1\ f_1, \texttt{bind}^{\mathrm{M}_2}\ m_2\ f_2) \leq \texttt{bind}^{\mathrm{W}_{rel}}\,(\theta_{rel}\,(m_1, m_2))\,(\theta_{rel} \circ (f_1, f_2))$$

2^{nd} extension (for exceptions)

$$\mathrm{W}^{\mathrm{Exc}}_{\mathrm{rel}}(A_1, A_2) = ((A_1 + E_1) \times (A_2 + E_2) \longrightarrow \mathbb{P}) \longrightarrow \mathbb{P}$$

$$\frac{\vdash m_1 \sim m_2 \ \{ \ w^m \ \} \qquad \forall a_1, a_2 \vdash f_1\, a_1 \sim f_2\, a_2 \ \left\{ w^f\ (a_1, a_2) \right\}}{\vdash \mathtt{bind}^{\mathrm{M}_1}\ m_1\ f_1 \sim \mathtt{bind}^{\mathrm{M}_2}\ m_2\ f_2 \ \left\{ \mathtt{bind}^{\mathrm{W}_{\mathrm{rel}}}\ w^m\ w^f \right\}}$$

$$
\begin{aligned}
&\mathtt{let}\ \mathtt{bind}^{\mathrm{W}^{\mathrm{Exc}}_{\mathrm{rel}}}\ w_m\ (w_{f_1} : A_1 \longrightarrow ((B_1 + E_1) \longrightarrow \mathbb{P}) \longrightarrow \mathbb{P}) \\
&\qquad\qquad\qquad (w_{f_2} : A_2 \longrightarrow ((B_2 + E_2) \longrightarrow \mathbb{P}) \longrightarrow \mathbb{P})\ w_f\ \varphi = \\
&\quad w_m\ (\lambda x : (A_1 + E_1) \times (A_2 + E_2). \\
&\qquad \mathtt{match}\ x\ \mathtt{with} \\
&\qquad \mid \mathrm{Inl}\ a_1, \mathrm{Inl}\ a_2 \longrightarrow w_f\ a_1\ a_2\ \varphi \\
&\qquad \mid \mathrm{Inr}\ e_1, \mathrm{Inr}\ e_2 \longrightarrow \varphi\ (\mathrm{Inr}\ e_1, \mathrm{Inr}\ e_2) \\
&\qquad \mid \mathrm{Inl}\ a_1, \mathrm{Inr}\ e_2 \longrightarrow w_{f_1}\ a_1\ (\lambda be.\ \varphi\ be\ (\mathrm{Inr}\ e_2)) \\
&\qquad \mid \mathrm{Inr}\ e_1, \mathrm{Inl}\ a_2 \longrightarrow w_{f_2}\ a_2\ (\lambda be.\ \varphi\ (\mathrm{Inr}\ e_1)\ be)\ )
\end{aligned}
$$

2nd extension is complex!

$$\mathrm{W}^{\mathrm{Exc}}_{\mathrm{rel}}(A_1, A_2) = ((A_1 + E_1) \times (A_2 + E_2) \longrightarrow \mathbb{P}) \longrightarrow \mathbb{P}$$

$$\Gamma \vDash c_1 \{w_1\} \sim c_2 \{w_2\} \mid w_{\mathrm{rel}} \quad = \quad \begin{pmatrix} \forall \gamma_1 : \Gamma_1, \theta_1(c_1\, \gamma_1) \leq w_1\, \gamma_1, \\ \forall \gamma_2 : \Gamma_2, \theta_2(c_2\, \gamma_2) \leq w_2\, \gamma_2, \\ \forall (\gamma_1, \gamma_2) : \Gamma_1 \times \Gamma_2, \theta_{\mathrm{rel}}(c_1\, \gamma_1, c_2\, \gamma_2) \leq w_{\mathrm{rel}}(\gamma_1, \gamma_2) \end{pmatrix}$$

We tame some of the complexity by switching to a *relational* dependent type theory (embedded in Coq)

The first relational program logic for catchable exceptions

Conclusions

Once we're completely done with the theory ...

... and **work out some more examples**

... **this could be a good fit for F*!**

EasyCrypt-style relational verification

- for an **actual programming language** with **dependent types** and tons of other goodies
- for **arbitrary** effects, relational specification monads, and relational monad morphisms

Verify your crypto proofs entirely in F*!