Prophecy Variables in Separation Logic

(Extending Iris with Prophecy Variables)

Ralf Jung, **Rodolphe Lepigre**, Gaurav Parthasarathy,
Marianna Rapoport, Amin Timany, Derek Dreyer, Bart Jacobs

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

What is Iris?! What are prophecy variables?!

The Iris framework:

- Higher-order concurrent separation logic framework in Coq (developed at MPI-SWS and elsewhere)
- Deployed in many verification projects (e.g., Rustbelt)
- Great for verifying tricky concurrent programs

Prophecy variables:

- Old idea introduced by Abadi and Lamport (1991)
- Lets you "peek into the future" of a program's execution when reasoning about an earlier step in the program
- **Never formally integrated into Hoare logic before!!!**

Our contribution: Prophecy variables for Iris

First integration of prophecy variables to Hoare logic!

- There was an informal treatment in Vafeiadis's thesis
- We discovered a flaw in proof of his key example (RDCSS)

Key idea of our approach:

- Model the right to resolve a prophecy as an ownable resource
- Leverage separation logic to easily ensure soundness

Implementation in Iris:

- Everything is formally verified in Coq
- Including key examples (RDCSS, Herlihy-Wing queues)

Part I – separation logic and prophecy variables

Resources and ownership

The notion of resource is pervasive:

- File system handle, memory location, permission…
- A resource should not be used concurrently (but this can sometimes be relaxed)
- In Iris they are expressed in terms of resource algebras (user-defined structures called cameras)

Examples of resource ownership:

- $\ell \mapsto v$ (exclusive ownership of a location ℓ)
- $\ell \overset{q}{\mapsto} v$ ("read only" (fractional) ownership of a location ℓ)
- $\mathrm{Proph}(p, v)$ (exclusive right to resolve prophecy p)

Separation logic (Iris base logic)

Separating conjunction $P * Q$:

- The resources are split in disjoint parts satisfying P and Q respectively
- $\ell_1 \mapsto v_1 * \ell_2 \mapsto v_2$ can only be valid if $\ell_1 \neq \ell_2$
- $\ell \mapsto v_1 * \ell \mapsto v_2$ is a contradiction
- The magic wand $P \mathrel{-\!\!*} Q$ is a form of implication

Other logical connectives for building Iris propositions:

- Conjunction $P \wedge Q$, disjunction $P \vee Q$...
- Universal and existential quantifiers
- Ownership of a resource $\boxed{a : M}^{\gamma}$ (and related connectives)
- Persistence modality $\square P$, later modality $\triangleright P$...

Hoare triples (and weakest preconditions)

We use Hoare triples $\{P\}\, e\, \{x.Q\}$ for specifications:

- The (potential) result of evaluating e is bound in Q
- Evaluation of e is safe in a state satisfying P
- If a value is reached the corresponding state and value satisfy Q
- Defined in terms of weakest preconditions:

$$\{P\}\, e\, \{x.Q\} \triangleq \square(P \twoheadrightarrow \mathsf{wp}\, e\, \{x.Q\})$$

Weakest preconditions are encoded using the base logic!

Example of specification: eager coin

We consider a specification for a "coin":

- A coin is only ever tossed once
- Reading its value always gives the same result

$$\{\mathsf{True}\}\,\texttt{new\_coin}()\,\{x.\ \exists c.\ \exists b.\ x = c \wedge \mathsf{Coin}(c, b)\}$$
$$\{\mathsf{Coin}(c, b)\}\,\texttt{read\_coin}(c)\,\{x.\ x = b \wedge \mathsf{Coin}(c, b)\}$$

Simple (eager) implementation:

$$\mathsf{Coin}(c, b) \triangleq c \mapsto b$$

$$\begin{aligned} \texttt{new\_coin}() &\triangleq \texttt{ref}(\mathit{nondet\_bool}()) \\ \texttt{read\_coin}(c) &\triangleq\ !c \end{aligned}$$

A lazy coin implementation

What if we want to flip the coin as late as possible?

$$\texttt{new\_coin}() \triangleq \texttt{ref}(\mathsf{None})$$

$$\begin{aligned}\texttt{read\_coin}(c) \triangleq\ & \texttt{match}\ !\, c\ \texttt{with} \\ & \quad \mathsf{Some}(b) \Rightarrow b \\ & \mid \mathsf{None} \quad\ \ \Rightarrow \texttt{let}\ b = \mathit{nondet\_bool}(); \\ & \qquad\qquad\qquad c \leftarrow \mathsf{Some}(b);\ b \\ & \texttt{end}\end{aligned}$$

To keep the same spec we need <u>prophecy variables</u>

Specification of the prophecy variables operations

Prophecy variables are used through two ghost code instructions

- `NewProph` creates a new prophecy variable
- `Resolve` p `to` v resolves prophecy variable p to value v

$$\{\mathsf{True}\}\,\texttt{NewProph}\,\{p.\ \exists v.\ \mathsf{Proph}(p, v)\}$$

$$\{\mathsf{Proph}(p, v)\}\,\texttt{Resolve}\,p\,\texttt{to}\,w\,\{x.\ x = () \wedge v = w\}$$

Principles of prophecy variables in speration logic:

- The future is ours
  $\mathsf{Proph}(p, v)$ gives exclusive right to resolve p
- We must fulfill our destiny
  A prophecy can only be resolved to the predicted value

Back to the lazy coin implementation

$$
\begin{aligned}
\texttt{new\_coin}() \triangleq\ & \texttt{let}\ r = \texttt{ref}(\mathsf{None}); \\
& \texttt{let}\ p = \texttt{NewProph}; \\
& \{val = r, proph = p\}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{read\_coin}(c) \triangleq\ & \texttt{match}\ !\, c.val\ \texttt{with} \\
& \quad \mathsf{Some}(b) \Rightarrow b \\
& \mid \mathsf{None} \quad\ \ \Rightarrow \texttt{let}\ b = nondet\_bool(); \\
& \qquad\qquad\qquad \texttt{Resolve}\ c.proph\ \texttt{to}\ b; \\
& \qquad\qquad\qquad c.val \leftarrow \mathsf{Some}(b);\ b \\
& \texttt{end}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Coin}(c, b) \triangleq\ & (c.val \mapsto \mathsf{Some}\ b) \\
\vee\ & (c.val \mapsto \mathsf{None} * \exists v. \\
& \quad \mathsf{Proph}(c.proph, v) * ValToBool(v) = b)
\end{aligned}
$$

Part II – weakest preconditions and adequacy

Model of weakest preconditions in Iris

Encoding of weakest preconditions (simplified):

$$\begin{aligned}
\mathsf{wp}\ e_1\ \{\Phi\} \triangleq\ & \text{if } e_1 \in \mathit{Val} \text{ then } \Phi(e_1) \text{ else} && \underline{\text{(return value)}}\\
& \forall \sigma_1.\, S(\sigma_1) \Rrightarrow\!\!\ast \\
& \quad \mathrm{reducible}(e_1, \sigma_1) \wedge && \underline{\text{(progress)}}\\
& \quad \forall e_2, \sigma_2, \vec{e}_f.\, ((e_1, \sigma_1) \to (e_2, \sigma_2, \vec{e}_f)) \Rrightarrow\!\!\ast \\
& \qquad S(\sigma_2) * \mathsf{wp}\ e_2\ \{\Phi\} * \mathop{\Huge\ast}_{e \in \vec{e}_f} \mathsf{wp}\ e\ \{\mathsf{True}\} && \underline{\text{(preservation)}}
\end{aligned}$$

$$S(\sigma) \triangleq \boxed{\bullet\, \sigma}^{\gamma_{\mathsf{heap}}} \qquad \underline{\text{(state interp.)}}$$

Some intuitions about the involved components:

- The state interpretation holds the state of the physical heap
- View shifts $P \Rrightarrow\!\!\ast\ Q$ allow updates to owned resources
- The actual definition uses the $\triangleright P$ modality to avoid circularity

Operational semantics: head reduction and observations

We extend reduction rules with observations:

$$
\begin{aligned}
(\overline{n} + \overline{m}, \sigma) &\rightarrow_{\mathsf{h}} (\overline{n + m}, \sigma, \epsilon, \epsilon) \\
(\mathtt{ref}(v), \sigma) &\rightarrow_{\mathsf{h}} (\ell, \sigma \uplus \{\ell \leftarrow v\}, \epsilon, \epsilon) \\
(\ell \leftarrow w, \sigma \uplus \{\ell \leftarrow v\}) &\rightarrow_{\mathsf{h}} (\ell, \sigma \uplus \{\ell \leftarrow w\}, \epsilon, \epsilon) \\
(\mathtt{fork}\ \{e\}, \sigma) &\rightarrow_{\mathsf{h}} ((), \sigma, e :: \epsilon, \epsilon) \\
(\mathtt{Resolve}\, p\, \mathtt{to}\, v, \sigma) &\rightarrow_{\mathsf{h}} ((), \sigma, \epsilon, (p, v) :: \epsilon) \\
(\mathtt{NewProph}, \sigma) &\rightarrow_{\mathsf{h}} (p, \sigma \uplus \{p\}, \epsilon, \epsilon)
\end{aligned}
$$

A couple of remarks:

- Observations are only recorded on resolutions
- The state σ now also records the prophecy variables in scope

Extension for prophecy variables

Encoding of weakest preconditions (simplified):

$$
\begin{array}{ll}
\mathsf{wp}\ e_1\ \{\Phi\} \triangleq \text{if } e_1 \in \mathit{Val} \text{ then } \Phi(e_1) \text{ else} & \underline{\text{(return value)}} \\
\quad \forall \sigma_1, \vec{\kappa}_1, \vec{\kappa}_2.\ S(\sigma_1, \vec{\kappa}_1 \mathbin{+\!\!+} \vec{\kappa}_2) \Rrightarrow\!\!\ast & \\
\qquad \mathrm{reducible}(e_1, \sigma_1) \land & \underline{\text{(progress)}} \\
\qquad \forall e_2, \sigma_2, \vec{e}_f.\ ((e_1, \sigma_1) \to (e_2, \sigma_2, \vec{e}_f, \vec{\kappa}_1)) \Rrightarrow\!\!\ast & \\
\qquad\quad S(\sigma_2, \vec{\kappa}_2) * \mathsf{wp}\ e_2\ \{\Phi\} * \mathop{\Huge\ast}_{e \in \vec{e}_f} \mathsf{wp}\ e\ \{\mathsf{True}\} & \underline{\text{(preservation)}}
\end{array}
$$

$$
\begin{array}{ll}
S(\sigma, \vec{\kappa}) \triangleq \boxed{\bullet\, \sigma.1}^{\gamma_{\mathbf{heap}}} * \ \exists \Pi.\ \boxed{\bullet\, \Pi}^{\gamma_{\mathbf{proph}}} \land \mathrm{dom}(\Pi) = \sigma.2 \land & \underline{\text{(state interp.)}} \\
\qquad \forall \{p \leftarrow vs\} \in \Pi.\ vs = \mathrm{filter}(p, \vec{\kappa}) &
\end{array}
$$

Some more intuitions about the involved components:

- The state interpretation holds observations that remain to be made
- Observations are removed from the list when taking steps

Statement of safety and adequacy

Safety with respect to a (pure) predicate:

$$\begin{aligned}
\mathit{Safe}_\phi(e_1) \triangleq \forall \vec{es}, \sigma, \vec{\kappa}.\ ([e_1], \varnothing) \to^*_{\mathsf{tp}} (e_2 :: \vec{es}, \sigma, \vec{\kappa}) \\
\Rightarrow \mathit{proper}_\phi(e_2, \sigma) \land \forall e \in \vec{es}.\ \mathit{proper}_{\mathsf{True}}(e, \sigma) \\
\mathit{proper}_\psi(e, \sigma) \triangleq (e \in \mathit{Val} \land \psi(e)) \lor \mathrm{reducible}(e, \sigma)
\end{aligned}$$

Theorem (adequacy). Let e be an expression and ϕ be a (pure) predicate. If $\mathsf{wp}\ e\,\{\phi\}$ is provable then $\mathit{Safe}_\phi(e)$.

Conclusion: what I did not show

Our prophecy variables support more features:

- Multi-resolution prophecies
- Resolution of prophecies on an atomic instructions
- Result value included in the prophecy resolutions

Our main motivations for prophecy variables in Iris:

- Logically atomic specifications (related to linearizability)
- Allow for much stronger proof rules
- Examples including RDCSS and Herlihy-Wing queues

Erasure theorem (elimination of ghost code)

Thanks! Questions?

(For more details: `https://iris-project.org`)