Prophecy Variables in Separation Logic
(Extending Iris with Prophecy Variables)

Ralf Jung, Rodolphe Lepigre, Gaurav Parthasarathy,
Marianna Rapoport, Amin Timany, Derek Dreyer, Bart Jacobs
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 \( \ell \))
- \( \ell \overset{q}{\mapsto} v \) (“read only” (fractional) ownership of a location \( \ell \))
- \( \text{Proph}(p, v) \) (**exclusive** right to resolve prophecy \( p \))
Separation logic (Iris base logic)

Separating conjunction $P \star Q$:
- The resources are split in disjoint parts satisfying $P$ and $Q$ respectively.
- $\ell_1 \mapsto v_1 \star \ell_2 \mapsto v_2$ can only be valid if $\ell_1 \neq \ell_2$
- $\ell \mapsto v_1 \star \ell \mapsto v_2$ is a contradiction.
- The magic wand $P \rightarrowarrow Q$ is a form of implication.

Other logical connectives for building Iris propositions:
- Conjunction $P \land Q$, disjunction $P \lor Q$...
- Universal and existential quantifiers.
- Ownership of a resource $\overline{a : M}^\gamma$ (and related connectives).
- Persistence modality $\square P$, later modality $\triangleright P$...
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:

$$\{P\} \ e \ {x.Q} \triangleq \Box(P \rightarrow wp \ e \ {x.Q})$$

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

\[
\{ \text{True} \} \text{new\_coin()} \{ x. \exists c. \exists b. x = c \land \text{Coin}(c, b) \} \\
\{ \text{Coin}(c, b) \} \text{read\_coin}(c) \{ x. x = b \land \text{Coin}(c, b) \}
\]

Simple (eager) implementation:

\[
\text{Coin}(c, b) \triangleq c \mapsto b \\
\text{new\_coin()} \triangleq \text{ref}(\text{nondet\_bool}()) \\
\text{read\_coin}(c) \triangleq !c
\]
A lazy coin implementation

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

\[
\text{new_coin()} \triangleq \text{ref}(\text{None})
\]

\[
\text{read_coin}(c) \triangleq \text{match } c \text{ with}
\]

\[
\begin{align*}
\text{Some}(b) & \Rightarrow b \\
\text{None} & \Rightarrow \text{let } b = \text{nondet_bool}(); \\
& \quad c \leftarrow \text{Some}(b); \ b \\
\end{align*}
\]

To keep the same spec we need prophecy variables
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*

\[
\{ \text{True} \} \text{NewProph} \{ p. \exists v. \text{Proph}(p, v) \} \\
\{ \text{Proph}(p, v) \} \text{Resolve } p \text{ to } w \{ x. x = () \land v = w \}
\]

Principles of prophecy variables in separation logic:

- **The future is ours**
  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

\[
\text{new_coin()} \triangleq \begin{array}{l}
\text{let } r = \text{ref}(\text{None}); \\
\quad \text{let } p = \text{NewProph}; \\
\quad \{\text{val} = r, \text{proph} = p\}
\end{array}
\]

\[
\text{read_coin}(c) \triangleq \begin{array}{l}
\text{match }!c.\text{val} \text{ with} \\
\quad \text{Some}(b) \Rightarrow b \\
\quad | \text{None} \Rightarrow \text{let } b = \text{nondet_bool}(); \\
\qquad \text{Resolve } c.\text{proph} \text{ to } b; \\
\qquad c.\text{val} \leftarrow \text{Some}(b); b
\end{array}
\]

\[
\text{Coin}(c, b) \triangleq (c.\text{val} \mapsto \text{Some } b) \\
\lor (c.\text{val} \mapsto \text{None} \land \exists v. \\
\text{Proph}(c.\text{proph}, v) \land \text{ValToBool}(v) = b)
\]
Part II – weakest preconditions and adequacy
Model of weakest preconditions in Iris

Encoding of weakest preconditions (simplified):

\[ wp \ e_1 \{ \Phi \} \triangleq \text{if } e_1 \in Val \text{ then } \Phi(e_1) \text{ else } \]
\[ \forall \sigma_1. \ S(\sigma_1) \implies * \]
\[ \text{reducible}(e_1, \sigma_1) \land \]
\[ \forall e_2, \sigma_2, \vec{e}_f. \ ((e_1, \sigma_1) \rightarrow (e_2, \sigma_2, \vec{e}_f)) \implies * \]
\[ S(\sigma_2) \ast wp \ e_2 \{ \Phi \} \ast \ast_{e \in \vec{e}_f} wp \ e \{ \text{True} \} \]

\[ S(\sigma) \triangleq \bullet^{\text{heap}}_{\sigma} \]

Some intuitions about the involved components:

- The state interpretation holds the state of the physical heap
- View shifts \( P \implies * Q \) 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:

\[(\overline{n} + \overline{m}, \sigma) \rightarrow h (\overline{n} + \overline{m}, \sigma, \epsilon, \epsilon)\]
\[(\text{ref}(v), \sigma) \rightarrow h (\ell, \sigma \sqcup \{\ell \leftarrow v\}, \epsilon, \epsilon)\]
\[(\ell \leftarrow w, \sigma \sqcup \{\ell \leftarrow v\}) \rightarrow h (\ell, \sigma \sqcup \{\ell \leftarrow w\}, \epsilon, \epsilon)\]
\[(\text{fork} \{e\}, \sigma) \rightarrow h ((), \sigma, e :: \epsilon, \epsilon)\]
\[(\text{Resolve } p \text{ to } v, \sigma) \rightarrow h ((), \sigma, \epsilon, (p, v) :: \epsilon)\]
\[(\text{NewProph}, \sigma) \rightarrow h (p, \sigma \sqcup \{p\}, \epsilon, \epsilon)\]

A couple of remarks:

- Observations are only recorded on resolutions
- The state \(\sigma\) now also records the prophecy variables in scope
Extension for prophecy variables

Encoding of weakest preconditions (simplified):

\[ \text{wp } e_1 \{ \Phi \} \triangleq \text{if } e_1 \in Val \text{ then } \Phi(e_1) \text{ else } \]
\[ \forall \sigma_1, \vec{k}_1, \vec{k}_2. S(\sigma_1, \vec{k}_1 \, \text{++} \, \vec{k}_2) \Rightarrow \]
reducible(e_1, \sigma_1) \land
\[ \forall e_2, \sigma_2, \vec{e}_f. ((e_1, \sigma_1) \rightarrow (e_2, \sigma_2, \vec{e}_f, \vec{k}_1)) \Rightarrow \]
\[ S(\sigma_2, \vec{k}_2) \ast \text{wp } e_2 \{ \Phi \} \ast \star_{e \in \vec{e}_f} \text{wp } e \{ \text{True} \} \]
\[ S(\sigma, \vec{k}) \triangleq [\bullet \sigma.1]^{\text{heap}} \ast \exists \Pi. [\bullet \Pi]^\gamma_{\text{proph}} \land \text{dom}(\Pi) = \sigma.2 \land \]
\[ \forall \{ p \leftarrow vs \} \in \Pi. vs = \text{filter}(p, \vec{k}) \]

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:

\[ \text{Safe}_\phi(e_1) \triangleq \forall \bar{e}s, \sigma, \bar{\kappa}. ([e_1], \emptyset) \rightarrow^* (e_2 :: \bar{e}s, \sigma, \bar{\kappa}) \]

\[ \Rightarrow \text{proper}_\phi(e_2, \sigma) \land \forall e \in \bar{e}s. \text{proper}_{\text{True}}(e, \sigma) \]

\[ \text{proper}_\psi(e, \sigma) \triangleq (e \in \text{Val} \land \psi(e)) \lor \text{reducible}(e, \sigma) \]

**Theorem (adequacy).** Let \( e \) be an expression and \( \phi \) be a (pure) predicate. If \( \text{wp} \ e \{ \phi \} \) is provable then \( \text{Safe}_\phi(e) \).
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)