Strong Logic for Weak Memory

Reasoning About Release-Acquire Consistency in Iris

Jan-Oliver Kaiser, Hoang-Hai Dang, Derek Dreyer, Ori Lahav, Viktor Vafeiadis

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

http://plv.mpi-sws.org/igps/

What is Iris?

Language-independent higher-order separation logic framework
with simple foundations for modular reasoning
about fine-grained concurrency in Coq

What is it Good for?

- The Rust Type System (Jung, Jourdan, Dreyer, Krebbers)
- Logical Relations (Krogh-Jespersen, Svendsen, Timany, Birkedal, Tassarotti, Jung, Krebbers)
- Object Capabilities (Swasey, Dreyer, Garg)
- Logical Atomicity (Krogh-Jespersen, Zhang, Jung)

Common theme: SC languages

Iris is very general — **but** it needs interleaving semantics

No support for weak memory?

We develop

interleaving semantics for C11 Release-Acquire

and derive

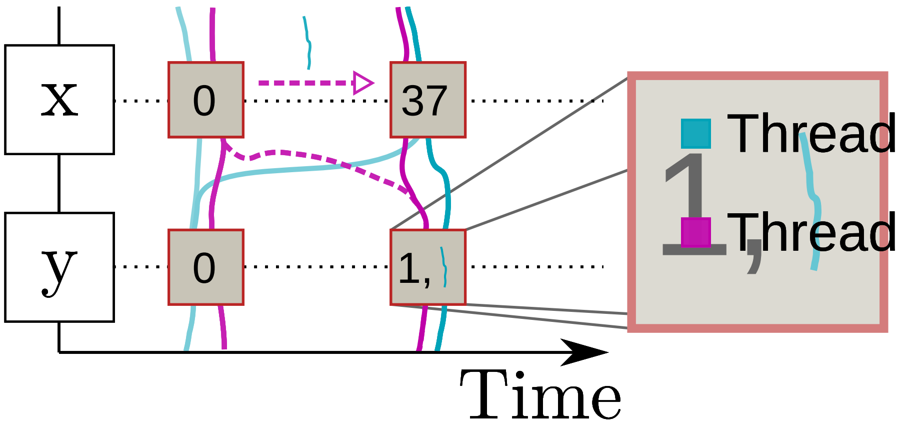
two major Release-Acquire program logics in Iris

Benefits of Using Iris for Weak Memory

- For free: separation, higher-order ghost state, impredicative invariants
- Mechanized soundness proofs at very high level of abstraction
- Mechanized examples: all examples of encoded logics, including RCU (PLDI' 2015)
- Mixing reasoning principles from different weak program logics

Operational Release-Acquire: Message Passing

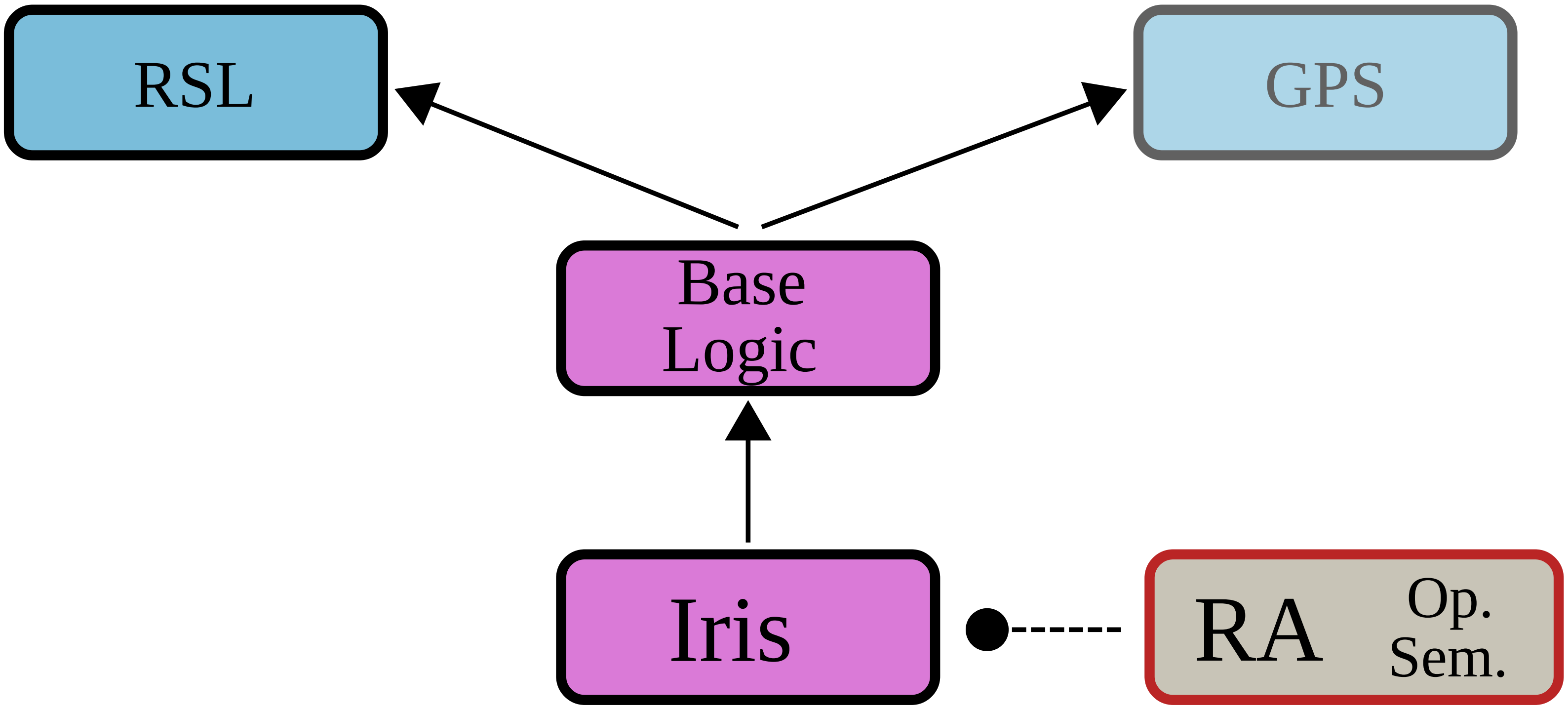
```
x := 0; y := 0
x[na] := 37  ||  repeat (!y[at])
y[at] := 1   ||  !x[na]
```



Support for Non-Atomic Accesses

- Built-in race detection for non-atomic accesses
- Allow mixing atomic and non-atomic accesses to same location
- (Almost) equivalent to C11

Roadmap



Iris: *Fictional Separation*

0. Have a monolithic, non-splittable resource r to be shared
1. Make a splittable ghost copy
   (by designing a PCM with the *right* separation structure)
2. Keep copy in sync with r using the Auth construction
   - unique authoritative copy marked with ●
   - splittable fragments marked with ◯
   - all ◯'s combine into ●
3. Use invariant to tie ● to original resource r
4. Derive rules to update ◯ in conjuction with ● and r

Iris: Heap with Fictional Separation

0. Monolithic resource: $\lfloor \sigma \rfloor$ — the physical state (heap)
1. We want exclusive, per-location fragments: $\mathsf{Loc} \xrightarrow{\text{fin}}_{\uplus} \mathsf{Value}$
2. Wrap it in Auth.

   $\ell \hookrightarrow v \triangleq \boxed{\circ\, [\ell := v]}$
3. Establish invariant: $\boxed{\exists \sigma. \lfloor \sigma \rfloor * \boxed{\bullet\, \sigma}}$
4. Derive all rules from these two:
   - $\boxed{\bullet\, \sigma} * \boxed{\circ\, [\ell := v]} \Rightarrow \square(\sigma(\ell) = v)$
   - $\boxed{\bullet\, \sigma} * \boxed{\circ\, [\ell := v]} \Rrightarrow \boxed{\bullet\, \sigma[\ell \mapsto w]} * \boxed{\circ\, [\ell := w]}$

The Base Logic: Fictional Separation of $\lfloor \sigma \rfloor$

- $\sigma : \left\{ msgs : \mathsf{Messages};\ views : \mathsf{ThreadId} \xrightarrow{\text{fin}} \mathsf{View};\ \ldots \right\}$
- Excl. history: $\mathrm{Hist}(\ell, h) \triangleq \boxed{\circ\, [\ell := h]}$ $(h : \mathcal{P}(\mathsf{Value} \times \mathsf{Time} \times \mathsf{View}))$
- Excl. views: $\mathrm{Seen}(\pi, V) \triangleq \boxed{\circ\, [\pi := V]}$
- One big invariant:
  $\boxed{\exists \sigma. \lfloor \sigma \rfloor * \boxed{\bullet\, \sigma.msgs} * \boxed{\bullet\, \sigma.views} * \ldots}$
- Relating thread views and history:
  - $\mathrm{init}(h, V) \triangleq \exists (v, \_, V_0) \in h. V \sqsupseteq V_0 \land v \in \mathsf{Value}$
  - $\mathrm{alloc}(h, V)$
  - $\mathrm{NA\text{-}safe}(h, V)$

Base Logic: Atomic Read

$$\textsc{Base-AT-Read}$$

$$\mathrm{PSCtx} \vdash \ \{\mathrm{Seen}(\pi, V) * \mathrm{Hist}(\ell, h) * \mathrm{init}(h, V)\}$$

$$!\ell_{[\mathrm{at}]}\,, \pi$$

$$\left\{\begin{array}{l} v.\ \ \exists V_1, V' \sqsupseteq V \sqcup V_1, t \geq V(\ell). \\ \quad \mathrm{Seen}(\pi, V') * \mathrm{Hist}(\ell, h) * (v, t, V_1) \in h \end{array}\right\}$$

Base Logic: Atomic Write

$$\textsc{Base-AT-Write}$$

$$\text{PSCtx} \vdash \ \{\text{Seen}(\pi, V) * \text{Hist}(\ell, h) * \text{alloc}(h, V)\}$$

$$\ell_{[\text{at}]} := v, \pi$$

$$\left\{ \begin{array}{l} \exists V \sqsupseteq V, t \geq V(\ell), h' = h \uplus \{(v, t, V')\}. \\ \quad \text{Seen}(\pi, V') * \text{Hist}(\ell, h') * \text{init}(h', V') \end{array} \right\}$$

Base Logic: Message Passing

```
           x := 0; y := 0
x[na] := 37  ||  repeat (!y[at])
y[at] := 1   ||  !x[na]
```

Invariants:

- $\text{Inv}_y(V_0) \triangleq$

  $\exists h.\ \text{Hist}(y, h) * (0, \_, V_0) \in h$

  $* \forall V_1, v_1 \neq 0. (v_1, \_, V_1) \in h \Rightarrow \exists V_{37} \sqsubseteq V_1.\ \boxed{\text{Inv}_x(V_{37})}$

- $\text{Inv}_x(V_{37}) \triangleq \boxed{\Diamond} * \text{Hist}(x, \{(37, \_, V_{37})\})$

Thread 1 proof outline:

$$\left\{\mathrm{Seen}(\pi, V_0) * \mathrm{Hist}(x, [(0, \_, V_x)]) * V_x \sqsubseteq V_0 * \boxed{\mathrm{Inv}_y(V_0)}\right\}$$

$$x_{[\mathrm{na}]} := 37$$

$$\{\exists V_{37} \sqsupseteq V_0.\ \mathrm{Seen}(\pi, V_{37}) * \mathrm{Hist}(x, [(37, \_, V_{37})])\}$$

$$\left\{\mathrm{Seen}(\pi, V_{37}) * \boxed{\mathrm{Inv}_x(V_{37})}\right\}$$

open Inv_y

$$\{\mathrm{Seen}(\pi, V_{37}) * \exists h.\ \mathrm{Hist}(y, h) * \ldots\}$$

$$y_{[\mathrm{at}]} := 1$$

$$\left\{\exists V_1 \sqsupseteq V_{37}.\ \mathrm{Seen}(\pi, V_1) * \mathrm{Hist}(y, h \uplus [(1, \_, V_1)]) * \boxed{\mathrm{Inv}_x(V_{37})}\right\}$$

$$\left\{\mathrm{Seen}(\pi, V_1) * \boxed{\mathrm{Inv}_y(V_0)}\right\}$$

Thread 2 proof outline:

$$\{\mathrm{Seen}(\pi, V_0) * \boxed{\mathrm{Inv}_y(V_0)} * \boxed{\Diamond}\}$$

repeat $y_{[\mathrm{at}]}$;

$$\{\exists V_1, V_{37}, V_2.\ V_2 \sqsupseteq V_1 \sqsupseteq V_{37} * \mathrm{Seen}(\pi, V_2) * \boxed{\mathrm{Inv}_x(V_{37})} * \boxed{\Diamond}\}$$

$$\{\mathrm{Seen}(\pi, V_2) * V_{37} \sqsubseteq V_2 * \mathrm{Hist}(x, [(37, \_, V_{37})])\}$$

$x_{[\mathrm{na}]}$

$$\{z.\ \mathrm{Seen}(\pi, V_2) * z = 37 * \mathrm{Hist}(x, [(37, \_, V_{37})])\}$$

Thread 2 proof outline:

$$\left\{\mathrm{Seen}(\pi, V_0) * \boxed{\mathrm{Inv}_y(V_0)} * \boxed{\diamond}\right\}$$

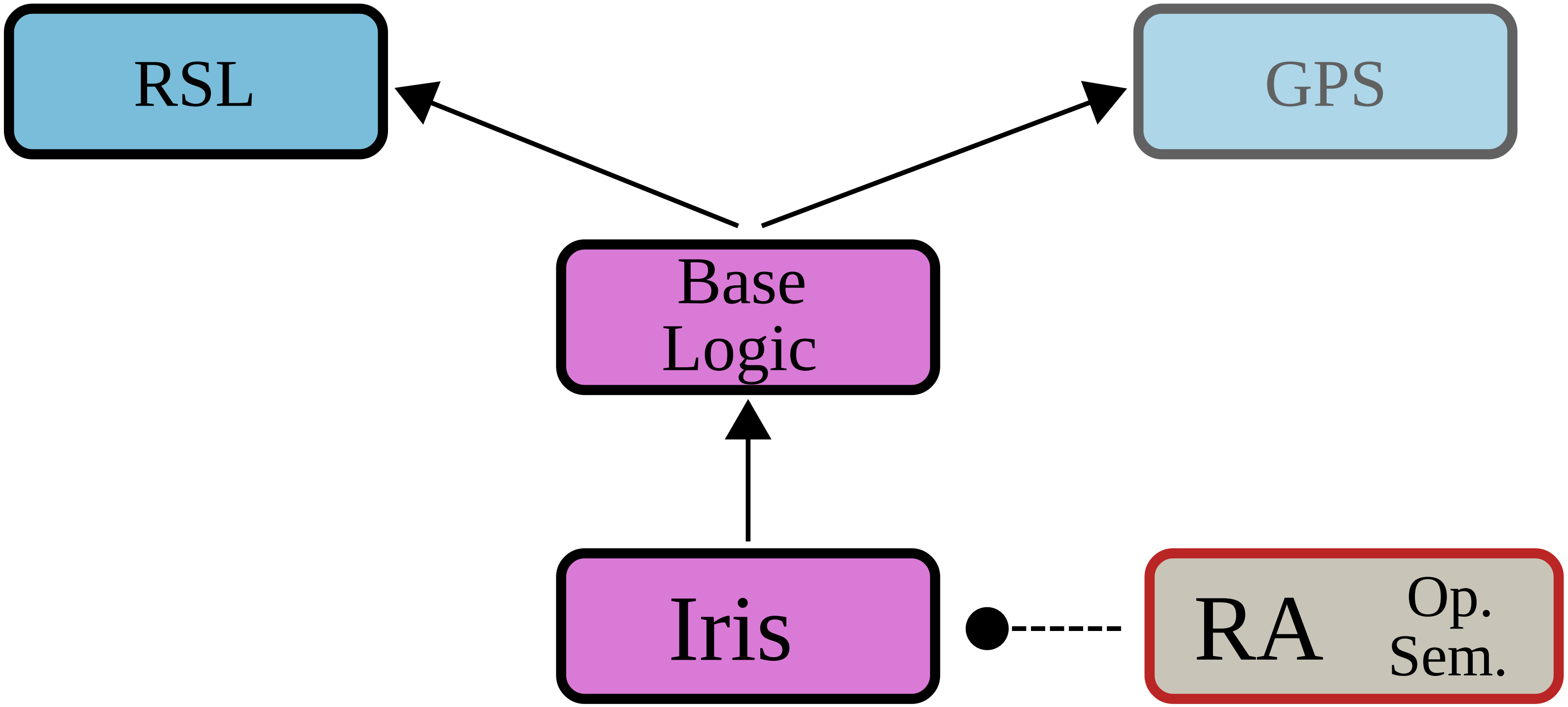
repeat $y_{[\mathrm{at}]}$;

$$\left\{\exists V_1, V_{37}, V_2.\ V_2 \sqsupseteq V_1 \sqsupseteq V_{37} * \mathrm{Seen}(\pi, V_2) * \boxed{\mathrm{Inv}_x(V_{37})} * \boxed{\diamond}\right\}$$

$$\{\mathrm{Seen}(\pi, V_2) * V_{37} \sqsubseteq V_2 * \mathrm{Hist}(x, [(37, \_, V_{37})])\}$$

$x_{[\mathrm{na}]}$

$$\{z.\ \mathrm{Seen}(\pi, V_2) * \underline{z = 37} * \mathrm{Hist}(x, [(37, \_, V_{37})])\}$$


RSL
GPS
Base
Logic
Iris
RA
Op.
Sem.

What is RSL?

RSL is a logic for message passing in Release-Acquire.

Main Ingredients: $\mathrm{Rel}(\ell, Q)$ and $\mathrm{Acq}(\ell, Q)$

$$\{\top\}\ \mathrm{alloc}\ \{\ell.\ \mathrm{Rel}(\ell, Q) * \mathrm{Acq}(\ell, Q)\}$$

$$\{\mathrm{Rel}(\ell, Q) * Q(v)\}\ \ell_{[\mathrm{at}]} := v\ \{\mathrm{Rel}(\ell, Q) * \mathrm{Init}(\ell)\}$$

$$\{\mathrm{Acq}(\ell, Q) * \mathrm{Init}(\ell)\}\ !\ell_{[\mathrm{at}]}\ \{v.\ \mathrm{Acq}(\ell, Q[v \mapsto \top]) * Q(v)\}$$

Message Passing in RSL

$$Q(v) = \begin{cases} x \hookrightarrow 37 & \text{if } v = 1 \\ \top & \text{ow.} \end{cases}$$

```
x := 0; y := 0
```

$\{x \hookrightarrow 0 * \mathrm{Rel}(y, Q)\}$	$\{\mathrm{Acq}(y, Q)\}$
`x[na] := 37`	`repeat (!y[at])`
$\{x \hookrightarrow 37 * \mathrm{Rel}(y, Q)\}$	$\{\mathrm{Acq}(y, \top) * x \hookrightarrow 37\}$
`y[at] := 1`	`!x[na]`
$\{\mathrm{Rel}(y, Q)\}$	$\{z.\ z = 37 * \ldots\}$

Challenge: How to Deal With Views?

Truth is relative to a thread's view.

Idea: Encode RSL assertions as *predicates on views.*

$$[\{P\}\ e\ \{Q\}]\,(V) \triangleq$$

$$\forall \pi.\ \{\mathrm{Seen}(\pi, V) * [P]\,(V)\}$$

$$e, \pi$$

$$\{\exists V' \sqsupseteq V.\, \mathrm{Seen}(\pi, V') * [Q]\,(V')\}$$

Frame

$$\{\mathrm{Seen}(\pi, V) * [P]\,(V)\}\ e, \pi\ \{\exists V' \sqsupseteq V.\, \mathrm{Seen}(\pi, V') * [Q]\,(V')\}$$

$$\frac{[\{P\}\ e\ \{Q\}]\,(V)}{[\{R * P\}\ e\ \{Q * R\}]\,(V)}$$

$$\{[R]\,(V) * \mathrm{Seen}(\pi, V) * [P]\,(V)\}\ e, \pi\ \{\exists V' \sqsupseteq V.\, \mathrm{Seen}(\pi, V') * [Q]\,(V') * [R]\,(V')\}$$

$$[R]\,(V) \overset{?}{\Rightarrow} [R]\,(V')$$

Use monotone predicates on views.

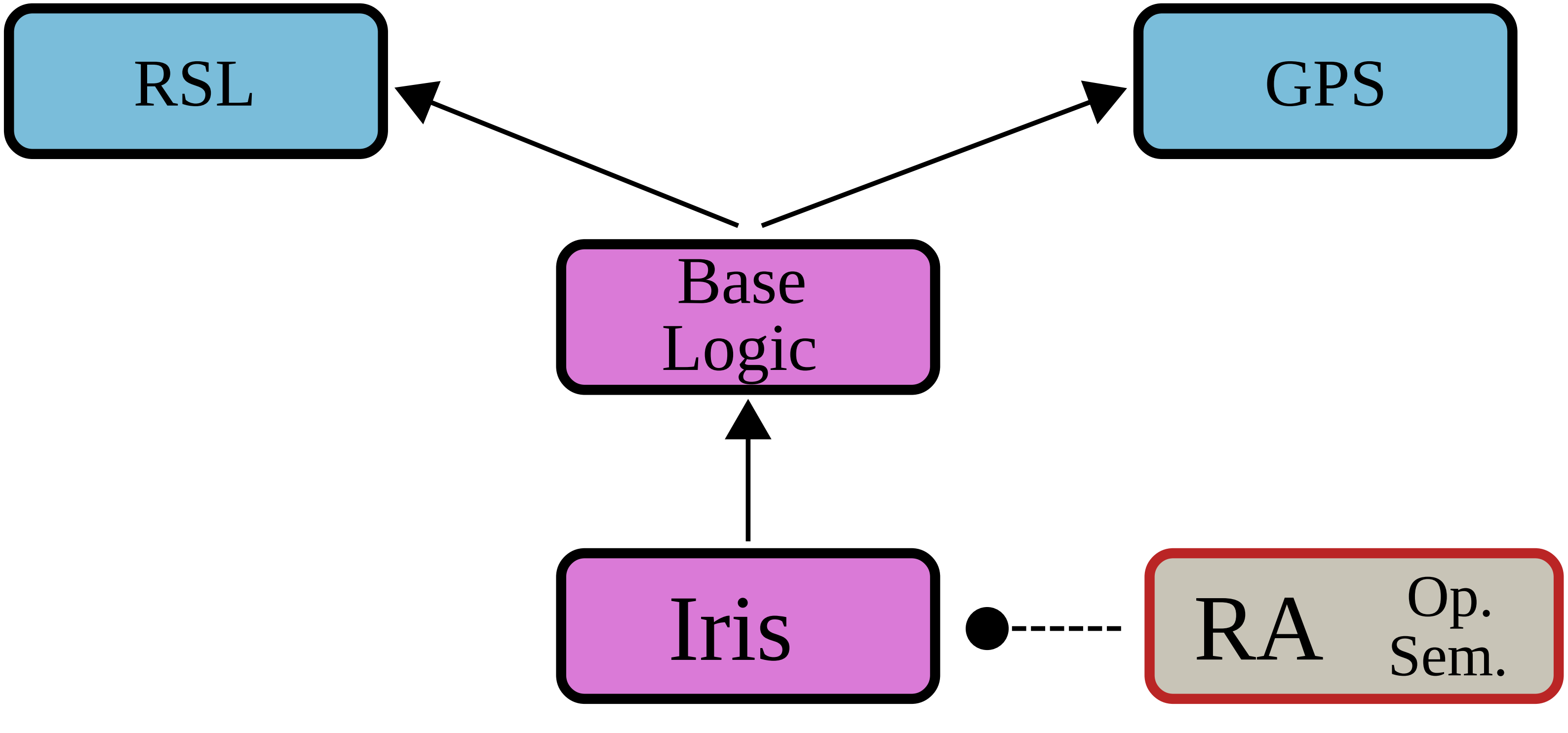
View-monotone predicates

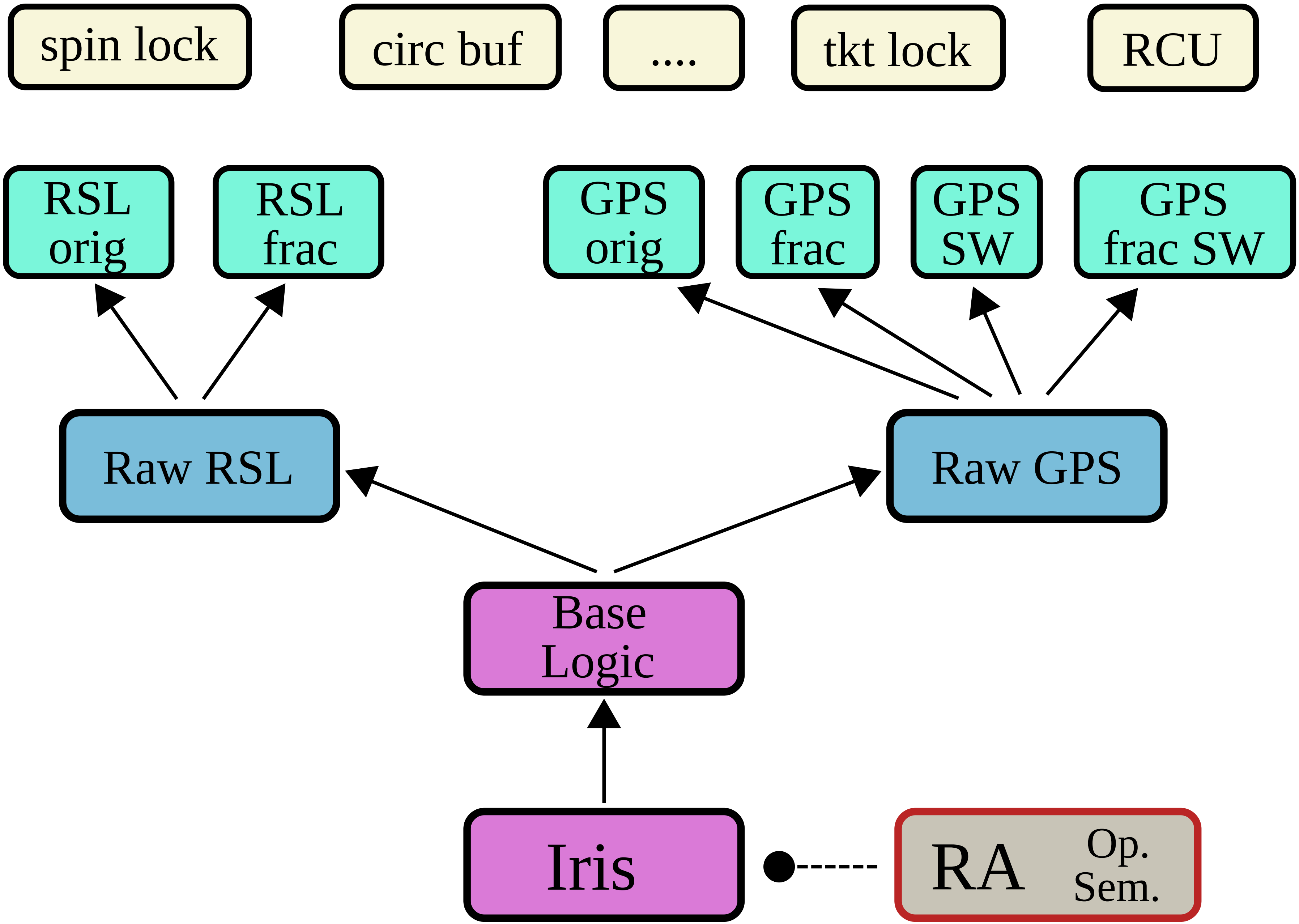
$$[\{P\}\ e\ \{Q\}]\,(V_0) \triangleq$$

$$\forall \pi, V \sqsupseteq V_0.\ \{\mathrm{Seen}(\pi, V) * [P]\,(V)\}$$

$$e, \pi$$

$$\{\exists V' \sqsupseteq V.\, \mathrm{Seen}(\pi, V') * [Q]\,(V')\}$$


RSL
GPS
Base
Logic
Iris
RA
Op.
Sem.


spin lock
circ buf
....
tkt lock
RCU
RSL orig
RSL frac
GPS orig
GPS frac
GPS SW
GPS frac SW
Raw RSL
Raw GPS
Base Logic
Iris
RA
Op. Sem.