KU LEUVEN

Reasoning about Object Capabilities with Logical Relations and Effect Parametricity

(EuroS&P 2016, Saarbrücken)

Dominique Devriese[1], Frank Piessens[1], Lars Birkedal[2]

[1] iMinds-DistriNet, KU Leuven,
[2] Aarhus University

December 2015

Example: browser ad sandboxing:

$$rnode \stackrel{\text{def}}{=} \mathtt{func}(node, d)\{\cdots\}$$

$$initWebPage \stackrel{\text{def}}{=} \mathtt{func}(document, ad)$$

$$\left\{ \begin{array}{l} \mathtt{let}\,(adNode = document.addChild(\text{“}ad\_div\text{”})) \\ \mathtt{let}\,(rAdNode = rnode(adNode, 0)) \\ ad.initialize(rAdNode) \end{array} \right\}$$

- Fine-grained privilege separation.
- Control authority of arbitrary, untrusted, untyped code.
  - Just restrict what it has access to.
- Low tech, low overhead.
  - No types/...
  - Standard OO techniques/patterns.
- High-level OO languages or low-level assembly
- Applications:
  - sandboxing
  - fault isolation
  - auditability
  - etc.

A *capability-safe* language:

- Private state encapsulation.
- Primitive I/O through non-public objects (like $document$).
- No global mutable state.

Examples:

- E, Joe-E, Emily, Newspeak etc.
- JavaScript 5 (strict mode, after proper initialisation)?

$$rnode \stackrel{\text{def}}{=} \texttt{func}(node, d)\{\cdots\}$$

$$initWebPage \stackrel{\text{def}}{=} \texttt{func}(document, ad) \left\{ \begin{array}{l} \texttt{let}\,(adNode = document.addChild(\text{``}ad\_div\text{''})) \\ \texttt{let}\,(rAdNode = rnode(adNode, 0)) \\ ad.initialize(rAdNode) \end{array} \right\}$$

Are we 100% sure?

- What does the language guarantee precisely? Is it really capability-safe? What does that mean?
- What to ensure precisely?
- What can we rely on precisely?

OCap community:

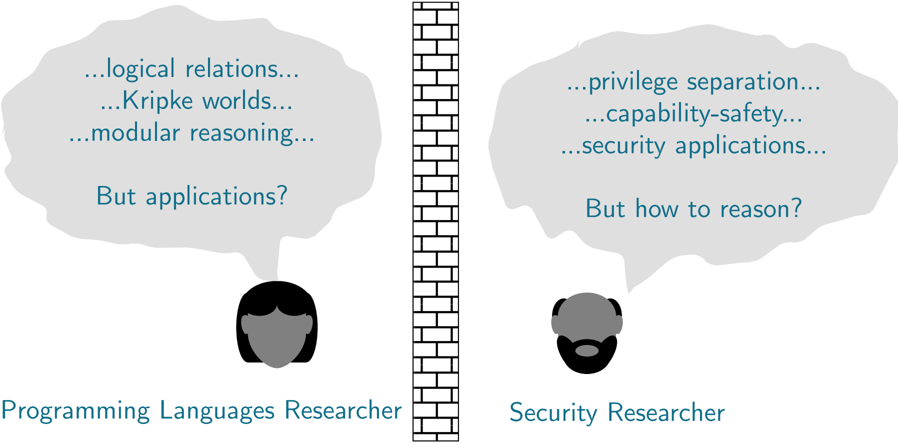
- Reference graph
- “No Authority Amplification”
- “Only Connectivity Begets Connectivity”

Problem:

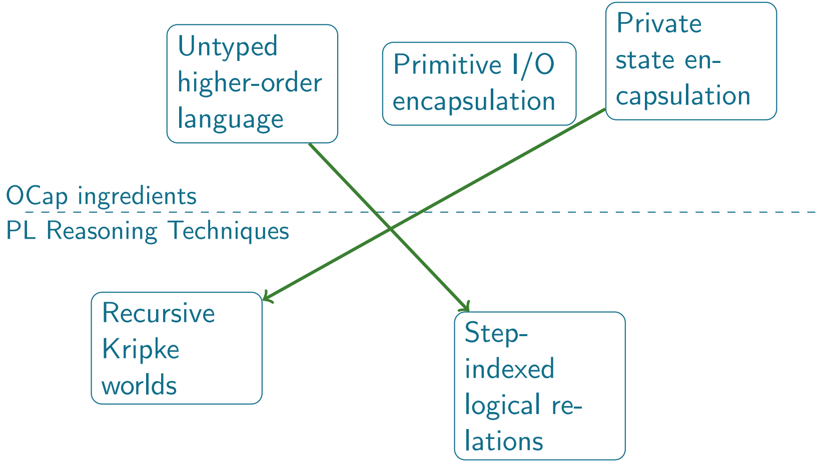
- Syntactic bound on authority.
- Ignores behavior.
- Necessary, but not sufficient!

What's the alternative?

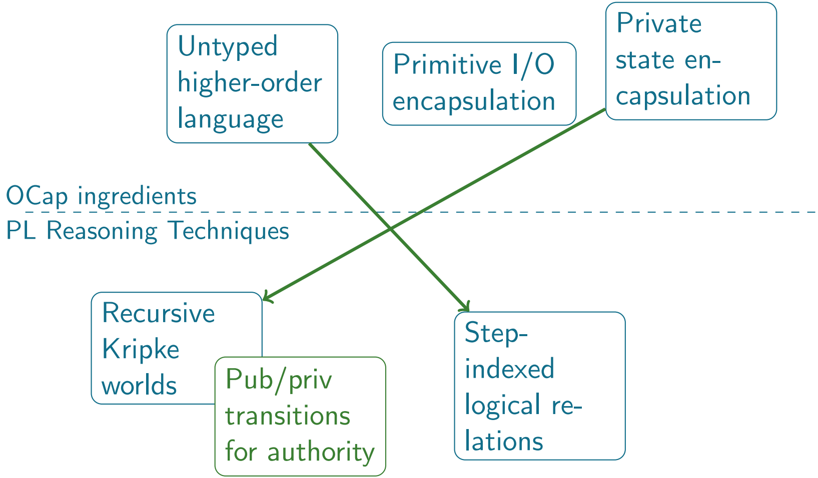
What's the alternative?



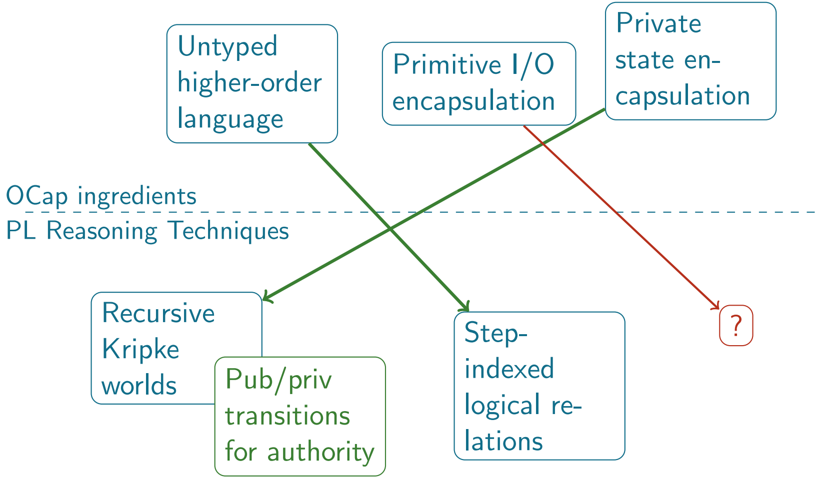
Our approach:



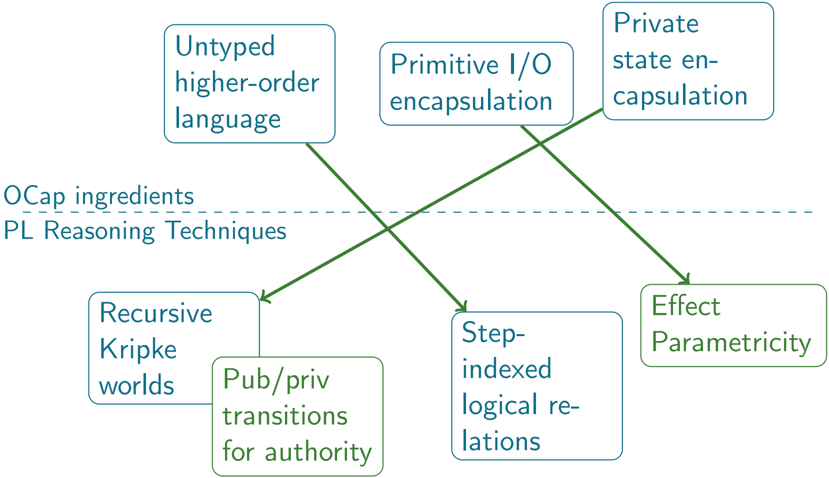
Our approach:



Our approach:



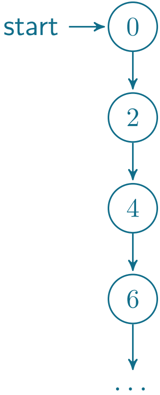
Our approach:



Encapsulation of private state

Register state machine to govern fresh data structure.

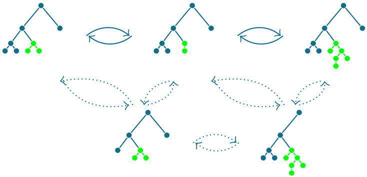
$$
\mathit{ticketDispenser} \stackrel{\text{def}}{=} \texttt{func}(\mathit{attacker})
\left\{
\begin{array}{l}
\texttt{let}(o = \texttt{ref}\ 0) \\
\ \texttt{let}\ (\mathit{dispTkt} = \texttt{func}()\{ \\
\quad \texttt{let}\ (v = \texttt{deref}\ o)\{o := v + 2; v\}\}) \\
\mathit{attacker}(\mathit{dispTkt}); \\
\ \texttt{deref}\ \ o
\end{array}
\right\}
$$



Public transitions: accessible under current authority.
Private transitions: potentially accessible by others.

$$rnode \stackrel{\text{def}}{=} \texttt{func}(node, d)\{\cdots\}$$

$$initWebPage \stackrel{\text{def}}{=} \texttt{func}(document, ad) \left\{ \begin{array}{l} \texttt{let}\,(adNode = document.addChild(\text{“}ad\_div\text{”})) \\ \texttt{let}\,(rAdNode = rnode(adNode, 0)) \\ ad.initialize(rAdNode) \end{array} \right\}$$



Effect interpretation: custom property about primitive effects

$\rho \in \mathcal{P}(Cap)$

$\mu \in \mathcal{P}(Val) \to \mathcal{P}(Expr)$

+ admissibility conditions...

Effect parametricity.

Theorem (Fundamental Theorem for λ_{JS})

If $\Gamma, \Sigma \vdash e$ *then for a valid effect interpretation* (μ, ρ) *and for all* n, γ *and* w *with* $(n, w) \in [\![\Sigma]\!]_{\mu,\rho}$ *and* $(n, \gamma) \in [\![\Gamma]\!]_{\mu,\rho}\ w$, *we have that* $(n, \gamma(e))$ *must be in* $\mathcal{E}[\mu\ \texttt{JSVal}_{\mu,\rho}]\ w$.

Conclusion

- Capability Safety is:
  - Private state encapsulation.
  - Absence of global state.
  - Primitive I/O encapsulation.
- Modular reasoning in cap-safe language:
  - Reference graph dynamics is not enough
  - Logical relations to the rescue.
- Some novel features:
  - Authority over shared data using public/private transitions.
  - Effect parametricity.
- (Not shown: relational version)

Future work

- Build effect interpretations into Kripke worlds?
- A program logic?
- Apply to full JavaScript?

Worlds:

$$\text{IslandName} \stackrel{\textsf{def}}{=} \mathbb{N}$$

$$W \stackrel{\textsf{def}}{=} \{w \in \text{IslandName} \hookrightarrow \text{Island} \mid \text{dom}(w) \textsf{ finite}\}$$

$$\text{Island} \stackrel{\textsf{def}}{=} \left\{ \begin{array}{r} \iota = (s, \phi, \phi^{\text{pub}}, H) \mid s \in \text{State} \wedge \phi \subseteq \text{State}^2 \wedge \\ H \in \text{State} \rightarrow \text{StorePred} \wedge \phi^{\text{pub}} \subseteq \phi \wedge \\ \phi, \phi^{\text{pub}} \textsf{ reflexive and transitive} \end{array} \right\}$$

$$\text{StorePred} \stackrel{\textsf{def}}{=} \{\psi \in \hat{W} \rightarrow_{mon,ne} \mathit{UPred}(\text{Store})\}$$

$$\mathit{roll} : \frac{1}{2} \cdot W \cong \hat{W}$$

Effect interpretations:

$$\rho : W \rightarrow_{mon,ne} UPred(Loc)$$
$$\mu : (W \rightarrow_{mon,ne} UPred(Val)) \rightarrow_{ne} (W \rightarrow_{ne} Pred(Cmd)).$$

$\texttt{JSVal}_{\mu,\rho}$ predicate:

$$\texttt{JSVal}_{\mu,\rho} : W \rightarrow_{mon,ne} UPred(Val)$$
$$\texttt{JSVal}_{\mu,\rho} \stackrel{\text{def}}{=} Cnst \cup \rho \cup \{\texttt{JSVal}_{\mu,\rho}\} \cup ([\texttt{JSVal}_{\mu,\rho}] \rightarrow \mu\ \texttt{JSVal}_{\mu,\rho})$$

Admissibility conditions for effect interpretation:

- A-PURE: If $(n, v) \in P\ w$ then $(n, v) \in \mu\ P\ w$
- A-BIND: If $(n, cmd) \in \mu\ P\ w$ and $(n', E\langle v\rangle) \in \mathcal{E}[\mu\ P']\ w'$ for all $n' \leq n$, $w' \sqsupseteq w$ and $(n', v) \in P\ w'$, then $(n, E\langle cmd\rangle) \in \mathcal{E}[\mu\ P']\ w$.
- A-ASSIGN: If $(n, v_1) \in \texttt{JSVal}_{\mu,\rho}\ w$ and $(n, v_2) \in \texttt{JSVal}_{\mu,\rho}\ w$, then $(n, v_1 = v_2) \in \mu\ \texttt{JSVal}_{\mu,\rho}\ w$.
- A-DEREF: If $(n, v) \in \texttt{JSVal}_{\mu,\rho}\ w$, $(n, \texttt{deref}\ v)$ must be in $\mu\ \texttt{JSVal}_{\mu,\rho}\ w$.
- A-REF: If $(n, v) \in \texttt{JSVal}_{\mu,\rho}\ w$, then $(n, \texttt{ref}\ v) \in \mu\ \texttt{JSVal}_{\mu,\rho}\ w$.