Capability-Flavoured Effects

GANG ZHOU, College of William and Mary
YAFENG WU, University of Virginia
TING YAN, Eaton Innovation Center
TIAN HE, University of Minnesota
CHENGDU HUANG, JOHN A. STANKOVIC, and TAREK F. ABDELZAHER, University of Virginia

Many modern applications require developers to build safe systems out of potentially unsafe components, but existing languages are insufficient in the techniques they provide for identifying untrustworthy or unsafe code. This report explores how capability-safety enables a low overhead effect-system that can reason about the authority of unannotated code. We demonstrate this with a capability calculus CC and give several scenarios where it helps developers make more informed choices about whether to trust code.

CCS Concepts: •**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

General Terms: Capabilities, effects

Additional Key Words and Phrases: Capabilities, effects

ACM Reference format:
Gang Zhou, Yafeng Wu, Ting Yan, Tian He, Chengdu Huang, John A. Stankovic, and Tarek F. Abdelzaher. 2010. Capability-Flavoured Effects. *ACM J. Comput. Cult. Herit.* 9, 4, Article 39 (March 2010), 9 pages.
DOI: 0000001.0000001

1 INTRODUCTION

Good software is distinguished from bad software by qualities such as security, maintainability, and performance. We are interested in how the design of a programming language and its type system can make it easier to write secure software.

There are different situations where we may not trust code. One example is in a development environment adhering to ideas of *code ownership*, wherein developers may exercise responsibility for specific components of the system [1]. When a developer writes code to interact with another component, they can make false assumptions about how it should be used. This can break correctness or leave components in a malconfigured state, putting the whole system at risk. Another setting involves applications which allow third-party plug-ins, some of which could be malicious. Such an example is a web mash-up, which brings together several disparate web applications into one unified service. In both cases, despite the presence of untrustworthy components, we want the system to function securely.

It is difficult to determine if a piece of code should be trusted, but a range of approaches can be taken about the issue. One is to *sandbox* untrusted code inside a virtual environment. If anything goes wrong, damage

DOI: 0000001.0000001

is theoretically limited to the virtual environment, but in practice there are many vulnerabilities [3, 7, 15, 19]. Verification gives a comprehensive analysis of the behaviour of code, but the techniques are heavyweight and developers must have a deep understanding of how they work in order to use them [5]. Furthermore, verification requires a complete specification of the system, which may be undefined or evolving throughout the development process. Lightweight analyses, such as type systems, are easy for the developer to use, but existing languages are insufficient in the tools they provide for detecting and isolating untrustworthy components [2, 18]. A qualitative approach might be taken, where software is developed according to best-practice guidelines. One such guideline is the *principle of least authority*: that software components should only have access to the information and resources necessary for their purpose [13]. For example, a logger module, which only needs to append to a file, should not be given arbitrary read-write access. Another is *privilege separation*, where the division of a program into components is informed by what resources are needed and how they are to be propagated [14]. This report focuses on the class of lightweight analyses, and in particular how type systems can be used to reject unsafe programs and put developers in a more informed position to make qualitative assessments.

One approach to privilege separation is the capability model. A *capability* is an unforgeable token granting its bearer permission to perform some operation [4]. For example, a system resource like a file or socket can only be used through a capability granting operations on it. Capabilities also encapsulate the source of *effects*, which describe intensional details about the way in which a program executes [10]. For example, a logger might `append` to a `File`, and so executing its code would incur a `File.append` effect. In the capability model, this requires the logger to possess a capability granting it the ability to append to that particular file.

Although the idea of a capability is an old one, there has been recent interest in its application to programming language design. Miller has identified the ways in which capabilities should proliferate to encourage *robust composition* — a set of ideas summarised by "only connectivity begets connectivity" [9]. As a result, a program's components are explicitly parameterised by the capabilities they may use; the only effects a component can incur are those for which it has been given a capability. Building on these ideas, Maffeis et. al. formalised the notion of a *capability-safe* language and showed a subset of Caja (a JavaScript implementation) is capability-safe [8]. Another capability-safe language, and one we use in this report, is Wyvern [11].

Effect systems were introduced by Lucassen and Gifford to determine what code can be safely parallelised [6]. They have also been applied to problems such as determining which functions might be invoked in a program [17] or determining which regions in memory may be accessed or updated [16]. Knowing what effects a piece of code might incur allows a developer to determine if code is trustworthy before executing it. This can be qualitatively assessed by comparing the static approximation of its effects to its expected least authority — a "logger" implementation which could write to a `Socket` is not to be trusted!

Despite these benefits, effect systems have seen little use in mainstream programming languages. Rytz et. al. believe the verbosity of their annotations is the main reason [12]. Successive works have focussed on reducing the developer overhead through techniques such as inference. Inference enables developers to rapidly prototype without annotations, incrementally adding them as their safety needed, but the benefit of capabilities for this has received less attention: because capabilities encapsulate the source of effects, and because capability-safety impose constraints on how they can propagate through the system, the effects of a piece of code can be safely approximated by inspecting what capabilities are passed into it. This is the key contribution of this report: capability-safety facilitates a lightweight, incremental effect discipline.

We begin by discussing preliminary concepts involving the formal definition of programming languages, effect systems, and Miller's capability model. Chapter 3 introduces the Operation Calculus `OC`, a typed lambda calculus with a simple notion of capabilities and their operations in which all code is effect-annotated. Relaxing this requirement, we then introduce the Capability Calculus `CC`, which permits the nesting of unannotated code inside annotated code in a controlled, capability-safe manner. A safe inference about the unannotated code can be made by inspecting the capabilities passed into it from its annotated surroundings. In chapter 4 we show how CC can model practical examples. We finish with a summary and a literature comparison.

2 BACKGROUND

This is the background.

3 CALCULI

4 APPLICATIONS

5 CONCLUSIONS

A OC PROOFS

LEMMA A.1 (OC CANONICAL FORMS). *Unless the rule used is ε-SUBSUME, the following are true:*

(1) *If* $\Gamma \vdash x : \tau$ `with` ε *then* $\varepsilon = \varnothing$.
(2) *If* $\Gamma \vdash v : \tau$ `with` ε *then* $\varepsilon = \varnothing$.
(3) *If* $\Gamma \vdash v : \{\bar{r}\}$ `with` ε *then* $v = r$ *and* $\{\bar{r}\} = \{r\}$.
(4) *If* $\Gamma \vdash v : \tau_1 \rightarrow_{\varepsilon'} \tau_2$ `with` ε *then* $v = \lambda x : \tau.e$.

PROOF.

(1) The only rule that applies to variables is ε-VAR which ascribes the type $\varnothing$.
(2) By definition a value is either a resource literal or a lambda. The only rules which can type values are ε-RESOURCE and ε-ABS. In the conclusions of both, $\varepsilon = \varnothing$.
(3) The only rule ascribing the type $\{\bar{r}\}$ is ε-RESOURCE. Its premises imply the result.
(4) The only rule ascribing the type $\tau_1 \rightarrow_{\varepsilon'} \tau_2$ is ε-ABS. Its premises imply the result.

□

THEOREM A.2 (OC PROGRESS). *If* $\Gamma \vdash e : \tau$ `with` ε *and* e *is not a value or variable, then* $e \longrightarrow e' \mid \varepsilon$*, for some* e', ε.

PROOF. By induction on $\Gamma \vdash e : \tau$ `with` ε.

Case: ε-VAR, ε-RESOURCE, or ε-ABS. Then e is a value or variable and the theorem statement holds vacuously.

Case: ε-APP. Then $e = e_1\ e_2$. If e_1 is not a value or variable it can be reduced $e_1 \longrightarrow e_1' \mid \varepsilon$ by inductive assumption, so $e_1\ e_2 \longrightarrow e_1'\ e_2 \mid \varepsilon$ by E-APP1. If $e_1 = v_1$ is a value and e_2 a non-value, then e_2 can be reduced $e_2 \longrightarrow e_2' \mid \varepsilon$ by inductive assumption, so $e_1\ e_2 \longrightarrow v_1\ e_2' \mid \varepsilon$ by E-APP2. Otherwise $e_1 = v_1$ and $e_2 = v_2$ are both values. By inversion on ε-APP and canonical forms, $\Gamma \vdash v_1 : \tau_2 \rightarrow_{\varepsilon'} \tau_3$ `with` $\varnothing$, and $v_1 = \lambda x : \tau_2.e_{body}$. Then $(\lambda x : \tau.e_{body})v_2 \longrightarrow [v_2/x]e_{body} \mid \varnothing$ by E-APP3.

Case: ε-OPERCALL. Then $e = e_1.\pi$. If e_1 is a non-value it can be reduced $e_1 \longrightarrow e_1' \mid \varepsilon$ by inductive assumption, so $e_1.\pi \longrightarrow e_1'.\pi \mid \varepsilon$ by E-OPERCALL1. Otherwise $e_1 = v_1$ is a value. By inversion on ε-OPERCALL and canonical forms, $\Gamma \vdash v_1 : \{r\}$ `with` $\{r.\pi\}$, and $v_1 = r$. Then $r.\pi \longrightarrow$ `unit` $\mid \{r.\pi\}$ by E-OPERCALL2.

Case: ε-SUBSUME. If e is a value or variable, the theorem holds vacuously. Otherwise by inversion on ε-SUBSUME, $\Gamma \vdash e : \tau'$ `with` ε', and $e \longrightarrow e' \mid \varepsilon$ by inductive assumption.

□

Lemma A.3 (OC Substitution). *If* $\Gamma, x : \tau' \vdash e : \tau$ with ε *and* $\Gamma \vdash v : \tau'$ with $\varnothing$ *then* $\Gamma \vdash [v/x]e : \tau$ with ε.

Proof. By induction on the derivation of $\Gamma, x : \tau' \vdash e : \tau$ with ε.

Case: ε-Var. Then $e = y$ is a variable. Either $y = x$ or $y \neq x$. Suppose $y = x$. By applying canonical Forms to the theorem assumption $\Gamma, x : \tau' \vdash e : \tau'$ with $\varnothing$, hence $\tau' = \tau$. $[v/x]y = [v/x]x = v$, and by assumption, $\Gamma \vdash v : \tau'$ with $\varnothing$, so $\Gamma \vdash [v/x]y : \tau$ with $\varnothing$.

Otherwise $y \neq x$. By applying canonical forms to the theorem assumption $\Gamma, x : \tau' \vdash y : \tau$ with $\varnothing$, so $y : \tau \in \Gamma$. Since $[v/x]y = y$, then $\Gamma \vdash y : \tau$ with $\varnothing$ by ε-Var.

Case: ε-Resource. Because $e = r$ is a resource literal then $\Gamma \vdash r : \{r\}$ with $\varnothing$ by canonical forms. By definition $[v/x]r = r$, so $\Gamma \vdash [v/x]r : \{\bar{r}\}$ with $\varnothing$.

Case: ε-App. By inversion $\Gamma, x : \tau' \vdash e_1 : \tau_2 \rightarrow_{\varepsilon_3} \tau_3$ with ε_A and $\Gamma, x : \tau' \vdash e_2 : \tau_2$ with ε_B, where $\varepsilon = \varepsilon_A \cup \varepsilon_B \cup \varepsilon_3$ and $\tau = \tau_3$. From inversion on ε-App and inductive assumption, $\Gamma \vdash [v/x]e_1 : \tau_2 \rightarrow_{\varepsilon_3} \tau_3$ with ε_A and $\Gamma \vdash [v/x]e_2 : \tau_2$ with ε_B. By ε-App $\Gamma \vdash ([v/x]e_1)([v/x]e_2) : \tau_3$ with $\varepsilon_A \cup \varepsilon_B \cup \varepsilon_3$. By simplifying and applying the definition of substitution, this is the same as $\Gamma \vdash [v/x](e_1\ e_2) : \tau$ with ε.

Case: ε-OperCall. By inversion $\Gamma, x : \tau' \vdash e_1 : \{\bar{r}\}$ with ε_1 and $\tau =$ Unit and $\varepsilon = \varepsilon_1 \cup \{r.\pi \mid r \in \bar{r}, \pi \in \Pi\}$. By inductive assumption, $\Gamma \vdash [v/x]e_1 : \{\bar{r}\}$ with ε_1. Then by ε-OperCall, $\Gamma \vdash ([v/x]e_1).\pi :$ Unit with $\varepsilon_1 \cup \{r.\pi \mid r.\pi \in \bar{r} \times \Pi\}$. By simplifying and applying the definition of substitution, this is the same as $\Gamma \vdash [v/x](e_1.\pi) : \tau$ with ε.

Case: ε-Subsume. By inversion, $\Gamma, x : \tau' \vdash e : \tau_2$ with ε_2, where $\tau_2 <: \tau$ and $\varepsilon_2 \subseteq \varepsilon$. By inductive hypothesis, $\Gamma \vdash [v/x]e : \tau_2$ with ε_2. Then $\Gamma \vdash [v/x]e : \tau$ with ε by ε-Subsume.

□

Theorem A.4 (OC Preservation). *If* $\Gamma \vdash e_A : \tau_A$ with ε_A *and* $e_A \longrightarrow e_B \mid \varepsilon$, *then* $\tau_B <: \tau_A$ *and* $\varepsilon_B \cup \varepsilon \subseteq \varepsilon_A$, *for some* $e_B, \varepsilon, \tau_B, \varepsilon_B$.

Proof. By induction on the derivation of $\Gamma \vdash e_A : \tau_A$ with ε_A and then the derivation of $e_A \longrightarrow e_B \mid \varepsilon$.

Case: ε-Var, ε-Resource, ε-Unit, ε-Abs. Then e_A is a value and cannot be reduced, so the theorem holds vacuously.

Case: ε-App. Then $e_A = e_1\ e_2$ and $\Gamma \vdash e_1 : \tau_2 \longrightarrow_{\varepsilon_3} \tau_3$ with ε_1 and $\Gamma \vdash e_2 : \tau_2$ with ε_2 and $\tau_B = \tau_3$ and $\varepsilon_A = \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$. In each case we choose $\tau_B = \tau_A$ and $\varepsilon_B \cup \varepsilon = \varepsilon_A$.

Subcase: E-App1. Then $e_1\ e_2 \longrightarrow e_1'\ e_2 \mid \varepsilon$. By inversion on E-App1, $e_1 \longrightarrow e_1' \mid \varepsilon$. By inductive hypothesis and ε-Subsume $\Gamma \vdash v_1 : \tau_2 \longrightarrow_{\varepsilon_3} \tau_3$ with ε_1. Then $\Gamma \vdash e_1'\ e_2 : \tau_3$ with $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$ by ε-App.

Subcase: E-App2. Then $e_1 = v_1$ is a value and $e_2 \longrightarrow e_2' \mid \varepsilon$. By inversion on E-App2, $e_2 \longrightarrow e_2' \mid \varepsilon$. By inductive hypothesis and ε-Subsume $\Gamma \vdash e_2' : \tau_2$ with ε_2. Then $\Gamma \vdash v_1\ e_2' : \tau_3$ with $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$ by ε-App.

Subcase: E-App3. Then $e_1 = \lambda x : \tau_2.e_{body}$ and $e_2 = v_2$ are values and $(\lambda x : \tau_2.e_{body})\ v_2 \longrightarrow [v_2/x]e_{body} \mid \varnothing$. By inversion on the rule ε-App used to type $\lambda x : \tau_2.e_{body}$, we know $\Gamma, x : \tau_2 \vdash e_{body} : \tau_3$ with ε_3. $e_1 = v_1$ and $e_2 = v_2$ are values, so $\varepsilon_1 = \varepsilon_2 = \varnothing$ by canonical forms . Then by the substitution lemma, $\Gamma \vdash [v_2/x]e_{body} : \tau_3$ with ε_3 and $\varepsilon_A = \varepsilon_B = \varepsilon$.

Case: ε-OperCall. Then $e_A = e_1.\pi$ and $\Gamma \vdash e_1 : \{\bar{r}\}$ with ε_1 and $\tau_A =$ Unit and $\varepsilon_A = \varepsilon_1 \cup \{r.\pi \mid r \in \bar{r}, \pi \in \Pi\}$.

Subcase: E-OperCall1. Then $e_1.\pi \longrightarrow e_1'.\pi \mid \varepsilon$. By inversion on E-OperCall1, $e_1 \longrightarrow e_1' \mid \varepsilon$. By inductive hypothesis and application of ε-Subsume, $\Gamma \vdash e_1' : \{\bar{r}\}$ with ε_1. Then $\Gamma \vdash e_1'.\pi : \{\bar{r}\}$ with $\varepsilon_1 \cup \{r.\pi \mid r \in \bar{r}, \pi \in \Pi\}$ by ε-OperCall.

Subcase: E-OperCall2. Then $e_1 = r$ is a resource literal and $r.\pi \longrightarrow$ `unit` $\mid \{r.\pi\}$. By canonical forms, $\varepsilon_1 = \varnothing$. By ε-Unit, $\Gamma \vdash$ `unit` : `Unit with` $\varnothing$. Therefore $\tau_B = \tau_A$ and $\varepsilon \cup \varepsilon_B = \{r.\pi\} = \varepsilon_A$. □

Theorem A.5 (OC Single-step Soundness). *If $\Gamma \vdash e_A : \tau_A$ with ε_A and e_A is not a value, then $e_A \longrightarrow e_B \mid \varepsilon$, where $\Gamma \vdash e_B : \tau_B$ with ε_B and $\tau_B <: \tau_A$ and $\varepsilon_B \cup \varepsilon \subseteq \varepsilon_A$, for some $e_B, \varepsilon, \tau_B, \varepsilon_B$.*

Proof. If e_A is not a value then the reduction exists by the progress theorem. The rest follows by the preservation theorem. □

Theorem A.6 (OC Multi-step Soundness). *If $\Gamma \vdash e_A : \tau_A$ with ε_A and $e_A \longrightarrow^* e_B \mid \varepsilon$, where $\Gamma \vdash e_B : \tau_B$ with ε_B and $\tau_B <: \tau_A$ and $\varepsilon_B \cup \varepsilon \subseteq \varepsilon_A$.*

Proof. By induction on the length of the multi-step reduction.

Case: Length 0. Then $e_A = e_B$ and $\tau_A = \tau_B$ and $\varepsilon = \varnothing$ and $\varepsilon_A = \varepsilon_B$.

Case: Length $n+1$. By inversion the multi-step can be split into a multi-step of length n, which is $e_A \longrightarrow^* e_C \mid \varepsilon'$, and a single-step of length 1, which is $e_C \longrightarrow e_B \mid \varepsilon''$, where $\varepsilon = \varepsilon' \cup \varepsilon''$. By inductive assumption and preservation theorem, $\Gamma \vdash e_C : \tau_C$ with ε_C and $\Gamma \vdash e_B : \tau_B$ with ε_B, where $\tau_C <: \tau_A$ and $\varepsilon_C \cup \varepsilon' \subseteq \varepsilon_A$. By single-step soundness, $\tau_B <: \tau_C$ and $\varepsilon_B \cup \varepsilon'' \subseteq \varepsilon_C$. Then by transitivity, $\tau_B <: \tau$ and $\varepsilon_B \cup \varepsilon' \cup \varepsilon'' = \varepsilon_B \cup \varepsilon \subseteq \varepsilon_A$. □

B CC PROOFS

Lemma B.1 (CC Canonical Forms). *Unless the rule used is ε-Subsume, the following are true:*

(1) *If $\hat{\Gamma} \vdash x : \hat{\tau}$ with ε then $\varepsilon = \varnothing$.*
(2) *If $\hat{\Gamma} \vdash \hat{v} : \hat{\tau}$ with ε then $\varepsilon = \varnothing$.*
(3) *If $\hat{\Gamma} \vdash \hat{v} : \{\bar{r}\}$ with ε then $\hat{v} = r$ and $\{\bar{r}\} = \{r\}$.*
(4) *If $\hat{\Gamma} \vdash \hat{v} : \hat{\tau}_1 \rightarrow_{\varepsilon'} \hat{\tau}_2$ with ε then $\hat{v} = \lambda x : \tau.\hat{e}$.*

Proof. Same as for OC. □

Theorem B.2 (CC Progress). *If $\hat{\Gamma} \vdash \hat{e} : \hat{\tau}$ with ε and $\hat{e}$ is not a value, then $\hat{e} \longrightarrow \hat{e}' \mid \varepsilon$, for some $\hat{e}', \varepsilon$.*

Proof. By induction on the derivation of $\hat{\Gamma} \vdash \hat{e} : \hat{\tau}$ with ε.

Case: ε-Module. Then $\hat{e} =$ `import`(ε_s) $x = \hat{e}_i$ `in` e. If $\hat{e}_i$ is a non-value then $\hat{e}_i \longrightarrow \hat{e}_i' \mid \varepsilon$ by inductive assumption and `import`(ε_s) $x = \hat{e}_i$ `in` $e \longrightarrow$ `import`(ε_s) $x = \hat{e}_i'$ `in` $e \mid \varepsilon$ by E-Module1. Otherwise $\hat{e}_i = \hat{v}_i$ is a value and `import`(ε_s) $x = \hat{v}_i$ `in` $e \longrightarrow [\hat{v}_i/x]\text{annot}(e, \varepsilon_s) \mid \varnothing$ by E-Module2. □

Lemma B.3 (CC Substitution). *If $\hat{\Gamma}, x : \hat{\tau}' \vdash \hat{e} : \hat{\tau}$ with ε and $\hat{\Gamma} \vdash \hat{v} : \hat{\tau}'$ with $\varnothing$ then $\hat{\Gamma} \vdash [\hat{v}/x]\hat{e}_A : \hat{\tau}$ with ε.*

Proof. By induction on the derivation of $\hat{\Gamma}, x : \hat{\tau}' \vdash \hat{e} : \hat{\tau}$ with ε.

Case: ε-Module. Then the following are true.

(1) $\hat{e} = \mathtt{import}(\varepsilon_s)\ x = \hat{e}_i\ \mathtt{in}\ e$
(2) $\hat{\Gamma}, y : \hat{\tau}' \vdash \hat{e}_i : \hat{\tau}_i\ \mathtt{with}\ \varepsilon_i$
(3) $y : \mathtt{erase}(\hat{\tau}_i) \vdash e : \tau$
(4) $\hat{\Gamma}, y : \hat{\tau}' \vdash \mathtt{import}(\varepsilon_s)\ x = \hat{e}_i\ \mathtt{in}\ e : \mathtt{annot}(\tau, \varepsilon_s)\ \mathtt{with}\ \varepsilon_s \cup \varepsilon_i$
(5) $\varepsilon_s = \mathtt{effects}(\hat{\tau}_i) \cup \mathtt{ho\text{-}effects}(\mathtt{annot}(\tau, \varnothing))$
(6) $\hat{\tau}_A = \mathtt{annot}(\tau, \varepsilon)$
(7) $\hat{\varepsilon}_A = \varepsilon_s \cup \varepsilon_i$

By applying inductive assumption to (2) $\hat{\Gamma} \vdash [\hat{v}/x]\hat{e}_i : \hat{\tau}_i\ \mathtt{with}\ \varepsilon_i$. Then by ε-Module $\hat{\Gamma} \vdash \mathtt{import}(\varepsilon_s)\ y = [\hat{v}/x]\hat{e}_i\ \mathtt{in}\ e : \mathtt{annot}(\tau_i, \varepsilon_s)\ \mathtt{with}\ \varepsilon_s \cup \varepsilon_i$. By definition of substitution, the form in this judgement is the same as $[\hat{v}/x]\hat{e}$. ◻

Lemma B.4 (CC Approximation 1). *If* $\mathtt{effects}(\hat{\tau}) \subseteq \varepsilon$ *and* $\mathtt{ho\text{-}safe}(\hat{\tau}, \varepsilon)$ *then* $\hat{\tau} <: \mathtt{annot}(\mathtt{erase}(\hat{\tau}), \varepsilon)$.

Lemma B.5 (CC Approximation 2). *If* $\mathtt{ho\text{-}effects}(\hat{\tau}) \subseteq \varepsilon$ *and* $\mathtt{safe}(\hat{\tau}, \varepsilon)$ *then* $\mathtt{annot}(\mathtt{erase}(\hat{\tau}), \varepsilon) <: \hat{\tau}$.

Proof. By simultaneous induction on derivations of safe and ho-safe.

Case: $\hat{\tau} = \{\bar{r}\}$ Then $\hat{\tau} = \mathtt{annot}(\mathtt{erase}(\hat{\tau}), \varepsilon)$ and the results for both lemmas hold immediately.

Case: $\hat{\tau} = \hat{\tau}_1 \rightarrow_{\varepsilon'} \hat{\tau}_2$, $\mathtt{effects}(\hat{\tau}) \subseteq \varepsilon$, $\mathtt{ho\text{-}safe}(\hat{\tau}, \varepsilon)$ It is sufficient to show $\hat{\tau}_2 <: \mathtt{annot}(\mathtt{erase}(\hat{\tau}_2), \varepsilon)$ and $\mathtt{annot}(\mathtt{erase}(\hat{\tau}_1), \varepsilon) <: \hat{\tau}_1$, because the result will hold by S-Effects. To achieve this we shall inductively apply lemma 1 to $\hat{\tau}_2$ and lemma 2 to $\hat{\tau}_1$.

From $\mathtt{effects}(\hat{\tau}) \subseteq \varepsilon$ we have $\mathtt{ho\text{-}effects}(\hat{\tau}_1) \cup \varepsilon' \cup \mathtt{effects}(\hat{\tau}_2) \subseteq \varepsilon$ and therefore $\mathtt{effects}(\hat{\tau}_2) \subseteq \varepsilon$. From $\mathtt{ho\text{-}safe}(\hat{\tau}, \varepsilon)$ we have $\mathtt{ho\text{-}safe}(\hat{\tau}_2, \varepsilon)$. Therefore we can apply lemma 1 to $\hat{\tau}_2$.

From $\mathtt{effects}(\hat{\tau}) \subseteq \varepsilon$ we have $\mathtt{ho\text{-}effects}(\hat{\tau}_1) \cup \varepsilon' \cup \mathtt{effects}(\hat{\tau}_2) \subseteq \varepsilon$ and therefore $\mathtt{ho\text{-}effects}(\hat{\tau}_1) \subseteq \varepsilon$. From $\mathtt{ho\text{-}safe}(\hat{\tau}, \varepsilon)$ we have $\mathtt{ho\text{-}safe}(\hat{\tau}_1, \varepsilon)$. Therefore we can apply lemma 2 to $\hat{\tau}_1$.

Case: $\hat{\tau} = \hat{\tau}_1 \rightarrow_{\varepsilon'} \hat{\tau}_2$, $\mathtt{ho\text{-}effects}(\hat{\tau}) \subseteq \varepsilon$, $\mathtt{safe}(\hat{\tau}, \varepsilon)$ It is sufficient to show $\mathtt{annot}(\mathtt{erase}(\hat{\tau}_2), \varepsilon) <: \hat{\tau}_2$ and $\hat{\tau}_1 <: \mathtt{annot}(\mathtt{erase}(\hat{\tau}_1), \varepsilon)$, because the result will hold by S-Effects. To achieve this we shall inductively apply lemma 2 to $\hat{\tau}_2$ and lemma 1 to $\hat{\tau}_1$.

From $\mathtt{ho\text{-}effects}(\hat{\tau}) \subseteq \varepsilon$ we have $\mathtt{effects}(\hat{\tau}_1) \cup \mathtt{ho\text{-}effects}(\hat{\tau}_2) \subseteq \varepsilon$ and therefore $\mathtt{ho\text{-}effects}(\hat{\tau}_2) \subseteq \varepsilon$. From $\mathtt{safe}(\hat{\tau}, \varepsilon)$ we have $\mathtt{safe}(\hat{\tau}_2, \varepsilon)$. Therefore we can apply lemma 2 to $\hat{\tau}_2$.

From $\mathtt{ho\text{-}effects}(\hat{\tau}) \subseteq \varepsilon$ we have $\mathtt{effects}(\hat{\tau}_1) \cup \mathtt{ho\text{-}effects}(\hat{\tau}_2) \subseteq \varepsilon$ and therefore $\mathtt{effects}(\hat{\tau}_1) \subseteq \varepsilon$. From $\mathtt{safe}(\hat{\tau}, \varepsilon)$ we have $\mathtt{ho\text{-}safe}(\hat{\tau}_1, \varepsilon)$. Therefore we can apply lemma 1 to $\hat{\tau}_1$.

◻

Lemma B.6 (CC Annotation). *If the following are true:*

(1) $\hat{\Gamma} \vdash \hat{v}_i : \hat{\tau}_i\ \mathtt{with}\ \varnothing$
(2) $\Gamma, y : \mathtt{erase}(\hat{\tau}_i) \vdash e : \tau$
(3) $\mathtt{effects}(\hat{\tau}_i) \cup \mathtt{ho\text{-}effects}(\mathtt{annot}(\tau, \varnothing)) \cup \mathtt{effects}(\mathtt{annot}(\Gamma, \varnothing)) \subseteq \varepsilon_s$
(4) $\mathtt{ho\text{-}safe}(\hat{\tau}_i, \varepsilon_s)$

Then $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau}_i \vdash \texttt{annot}(e, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with ε_s.

Proof. By induction on the derivation of $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash e : \tau$. When applying the inductive assumption, e, τ, and Γ may vary, but the other variables are fixed.

Case: T-Var. Then $e = x$ and $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash x : \tau$. Either $x = y$ or $x \neq y$.

Subcase 1: $x = y$. Then $y : \texttt{erase}(\hat{\tau}_i) \vdash y : \tau$ so $\tau = \texttt{erase}(\hat{\tau}_i)$. By ε-Var, $y : \hat{\tau}_i \vdash x : \hat{\tau}_i$ with $\varnothing$. By definition $\texttt{annot}(x, \varepsilon_s) = x$, so (5) $y : \hat{\tau}_i \vdash \texttt{annot}(x, \varepsilon_s) : \hat{\tau}_i$ with $\varnothing$. By (3) and (4) we know $\texttt{effects}(\hat{\tau}_i) \subseteq \varepsilon_s$ and $\texttt{ho-safe}(\hat{\tau}_i, \varepsilon_s)$. By the approximation lemma, $\hat{\tau}_i <: \texttt{annot}(\texttt{erase}(\hat{\tau}_i), \varepsilon_s)$. We know $\texttt{erase}(\hat{\tau}_i) = \tau$, so this judgement can be rewritten as $\hat{\tau}_i <: \texttt{annot}(\tau, \varepsilon_s)$. From this we can use ε-Subsume to narrow the type of (5) and widen the approximate effects of (5) from $\varnothing$ to ε_s, giving $y : \hat{\tau}_i \vdash \texttt{annot}(x, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with ε_s. Finally, by widening the context, $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), \hat{\tau}_i \vdash \texttt{annot}(x, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with ε_s.

Subcase 2: $x \neq y$. Because $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash x : \tau$ and $x \neq y$ then $x : \tau \in \Gamma$. Then $x : \texttt{annot}(\tau, \varepsilon_s) \in \texttt{annot}(\Gamma, \varepsilon_s)$ so $\texttt{annot}(\Gamma, \varepsilon_s) \vdash x : \texttt{annot}(\tau, \varepsilon_s)$ with $\varnothing$ by ε-Var. By definition $\texttt{annot}(x, \varepsilon_s) = x$, so $\texttt{annot}(\Gamma, \varepsilon_s) \vdash \texttt{annot}(x, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with $\varnothing$. Applying ε-Subsume gives $\texttt{annot}(\Gamma, \varepsilon_s) \vdash \texttt{annot}(x, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with ε_s. By widening the context $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau}_i \vdash \texttt{annot}(\tau, \varepsilon_s)$ with ε'.

Case: T-Resource. Then $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash r : \{r\}$. By ε-Resource, $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon), y : \hat{\tau}_i \vdash r : \{r\}$ with $\varnothing$. Applying definitions, $\texttt{annot}(r, \varepsilon) = r$ and $\texttt{annot}(\{r\}, \varepsilon_s) = \{r\}$, so this judgement can be rewritten as $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon), y : \hat{\tau}_i \vdash \texttt{annot}(e, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with $\varnothing$. By ε-Subsume, $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau}_i \vdash \texttt{annot}(e, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with ε_s.

Case: T-Abs. Then $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash \lambda x : \tau_2.e_{body} : \tau_2 \to \tau_3$. Applying definitions, (5) $\texttt{annot}(e, \varepsilon_s) = \texttt{annot}(\lambda x : \tau_2.e_{body}, \varepsilon_s) = \lambda x : \texttt{annot}(\tau_2, \varepsilon_s).\texttt{annot}(e_{body}, \varepsilon_s)$ and $\texttt{annot}(\tau, \varepsilon_s) = \texttt{annot}(\tau_2 \to \tau_3, \varepsilon_s) = \texttt{annot}(\tau_2, \varepsilon_s) \to_{\varepsilon_s} \texttt{annot}(\tau_3, \varepsilon_s)$. By inversion on T-Abs, we get the sub-derivation (6) $\Gamma, y : \texttt{erase}(\hat{\tau}_i), x : \tau_2 \vdash e_{body} : \tau_2$. We shall apply the inductive assumption to this judgement with an unannotated context consisting of $\Gamma, x : \tau_2$. To be a valid application of the lemma, it is required that $\texttt{effects}(\texttt{annot}(\Gamma, x : \tau_2, \varnothing) \subseteq \varepsilon_s$. We already know $\texttt{effects}(\texttt{annot}(\Gamma, \varnothing)) \subseteq \varepsilon_s$ by assumption (3). Also by assumption (3), $\texttt{ho-effects}(\texttt{annot}(\tau_2 \to \tau_3, \varnothing)) \subseteq \varepsilon_s$; then by definition of ho-effects, $\texttt{effects}(\texttt{annot}(\tau_2, \varnothing)) \subseteq \texttt{ho-effects}(\texttt{annot}(\tau_2 \to \tau_3, \varnothing))$, so $\texttt{effects}(\texttt{annot}(x : \tau_2,))\varepsilon_s) \subseteq \varepsilon_s$ by transitivity. Then by applying the inductive assumption to (6), $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), \texttt{annot}(x : \tau_2, \varepsilon_s), y : \hat{\tau}_i \vdash \texttt{annot}(e_{body}, \varepsilon_s) : \texttt{annot}(\tau_3, \varepsilon_s)$ with ε_s. By ε-Abs, $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau}_i \vdash \lambda x : \texttt{annot}(\hat{\tau}_2, \varepsilon_s).\texttt{annot}(e_{body}, \varepsilon_s) : \texttt{annot}(\hat{\tau}_2, \varepsilon_s) \to_{\varepsilon_s} \texttt{annot}(\hat{\tau}_3, \varepsilon_s)$ with $\varnothing$. By applying the identities from (5), this judgement can be rewritten as $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau}_i \vdash \texttt{annot}(e, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with $\varnothing$. Finally, by applying ε-Subsume, $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau}_i \vdash \texttt{annot}(e, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with ε_s.

Case: T-App. Then $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash e_1\ e_2 : \tau_3$ and by inversion $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash e_1 : \tau_2 \to \tau_3$ and $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash e_2 : \tau_2$. By applying the inductive assumption to these judgements, $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau}_i \vdash \texttt{annot}(e_1, \varepsilon_2) : \texttt{annot}(\tau_2, \varepsilon_s) \to_{\varepsilon_s} \texttt{annot}(\tau_3, \varepsilon_s)$ with ε_s and $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau} \vdash \texttt{annot}(e_2, \varepsilon_s) : \texttt{annot}(\tau_2, \varepsilon_s)$ with ε_s. Then by ε-App, we get $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau} \vdash \texttt{annot}(e_1, \varepsilon_s)\ \texttt{annot}(e_2, \varepsilon_s) : \texttt{annot}(\tau_3, \varepsilon)$ with ε. Unfolding the definition of annot , this judgement can be rewritten as $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau} \vdash \texttt{annot}(e_1\ e_2, \varepsilon_s) : \texttt{annot}(\tau_3, \varepsilon)$ with ε. Finally, because $e = e_1\ e_2$ and $\tau = \tau_3$, this is the same as $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon_s), y : \hat{\tau} \vdash \texttt{annot}(e, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon)$ with ε.

Case: T-OperCall. Then $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash e_1.\pi : \texttt{Unit}$. By inversion we get the sub-derivation $\Gamma, y : \texttt{erase}(\hat{\tau}_i) \vdash e_1 : \{\bar{r}\}$. Applying the inductive assumption, $\hat{\Gamma}, \texttt{annot}(\Gamma, \varepsilon), y : \hat{\tau}_i \vdash \texttt{annot}(e_1, \varepsilon_s) : \texttt{annot}(\{\bar{r}\}, \varepsilon_s)$ with ε_s. By definition, $\texttt{annot}(\{\bar{r}\}, \varepsilon_s) = \{\bar{r}\}$, so this judgement can be rewritten as $\hat{\Gamma}, \texttt{annot}(\Gamma, \varnothing), y : \hat{\tau}_i \vdash e_1 : \{\bar{r}\}$ with ε_s. By

ε-OperCall, $\hat{\Gamma}, \texttt{annot}(\Gamma, \varnothing), y : \hat{\tau} \vdash \texttt{annot}(e_1.\pi, \varepsilon_s) : \{\bar{r}\}$ with $\varepsilon_s \cup \{\bar{r}.\pi\}$. All that remains is to show $\{\bar{r}.\pi\} \subseteq \varepsilon$. We shall do this by considering which subcontext left of the turnstile is capturing $\{\bar{r}\}$. Technically, $\hat{\Gamma}$ may not have a binding for every $r \in \bar{r}$: the judgement for e_1 might be derived using S-Resources and ε-Subsume. However, at least one binding for some $r \in \bar{r}$ must be present in $\hat{\Gamma}$ to get the original typing judgement being subsumed, so we shall assume without loss of generality that $\hat{\Gamma}$ contains a binding for every $r \in \bar{r}$.

Subcase 1: $\{\bar{r}\} = \hat{\tau}$. By assumption (3), $\texttt{effects}(\hat{\tau}) \subseteq \varepsilon_s$, so $\bar{r}.\pi \subseteq \{r.\pi \mid r \in \bar{r}, \pi \in \Pi\} = \texttt{effects}(\{\bar{r}\}) \subseteq \varepsilon_s$.

Subcase 2: $r : \{\bar{r}\} \in \texttt{annot}(\Gamma, \varepsilon_s)$. Then $\bar{r}.\pi \in \texttt{effects}(\{\bar{r}\}) \subseteq \texttt{effects}(\texttt{annot}(\Gamma, \varnothing))$, and by assumption (3) $\texttt{effects}(\texttt{annot}(\Gamma, \varnothing)) \subseteq \varepsilon_s$, so $\bar{r}.\pi \in \varepsilon_s$.

Subcase 3: $r : \{\bar{r}\} \in \hat{\Gamma}$. Because $\Gamma, y : \texttt{erase}(\hat{\tau}) \vdash e_1 : \{\bar{r}\}$, then $\bar{r} \in \Gamma$ or $r = \tau$. If $r \in \texttt{annot}(\Gamma, \varnothing)$ then subcase 2 holds. Else $r = \texttt{erase}(\hat{\tau})$. Because $\hat{\tau} = \{\bar{r}\}$, then $\texttt{erase}(\{\bar{r}\}) = \{\bar{r}\}$, so $\hat{\tau} = \tau$; therefore subcase 1 holds. □

Theorem B.7 (CC Preservation). *If $\hat{\Gamma} \vdash \hat{e}_A : \hat{\tau}_A$ with ε_A and $\hat{e}_A \longrightarrow \hat{e}_B \mid \varepsilon$, then $\hat{\Gamma} \vdash \hat{e}_B : \hat{\tau}_B$ with ε_B, where $\hat{e}_B <: \hat{e}_A$ and $\varepsilon \cup \varepsilon_B \subseteq \varepsilon_A$, for some $\hat{e}_B, \varepsilon, \hat{\tau}_B, \varepsilon_B$.*

Proof. By induction on the derivation of $\hat{\Gamma} \vdash \hat{e}_A : \hat{\tau}_A$ with ε_A and then the derivation of $\hat{e}_A \longrightarrow \hat{e}_B \mid \varepsilon$.

Case: ε-Import. Then by inversion on the rules used, the following are true:

(1) $\hat{e}_A = \texttt{import}(\varepsilon_s)\ x = \hat{v}_i$ in e
(2) $x : \texttt{erase}(\hat{\tau}_i) \vdash e : \tau$
(3) $\hat{\Gamma} \vdash \hat{e}_i : \hat{\tau}_i$ with ε_1
(4) $\hat{\Gamma} \vdash \hat{e}_A : \texttt{annot}(\tau, \varepsilon_s)$ with $\varepsilon_s \cup \varepsilon_1$
(5) $\texttt{effects}(\hat{\tau}_i) \cup \texttt{ho-effects}(\texttt{annot}(\tau, \varnothing)) \subseteq \varepsilon_s$
(6) $\texttt{ho-safe}(\hat{\tau}_i, \varepsilon_s)$

Subcase 1: E-Import1. Then $\texttt{import}(\varepsilon_s)\ x = \hat{e}_i$ in $e \longrightarrow \texttt{import}(\varepsilon_s)\ x = \hat{e}_i'$ in $e \mid \varepsilon$ and by inversion, $\hat{e}_i \longrightarrow \hat{e}_i' \mid \varepsilon$. By inductive assumption and subsumption, $\hat{\Gamma} \vdash \hat{e}_i' : \hat{\tau}_i'$ with ε_1. Then by ε-Import, $\hat{\Gamma} \vdash \texttt{import}(\varepsilon_s)\ x = \hat{e}_i'$ in $e : \texttt{annot}(\tau, \varepsilon_s)$ with ε_s.

Subcase 2: E-Import2. Then $\hat{e}_i = \hat{v}_i$ is a value and $\varepsilon_1 = \varnothing$ by canonical forms. Apply the annotation lemma with $\Gamma = \varnothing$ to get $\hat{\Gamma}, x : \hat{\tau}_i \vdash \texttt{annot}(e, \varepsilon_s) : \texttt{annot}(\tau, \varepsilon_s)$ with ε_s. From assumption (4) and canonical forms we have $\hat{\Gamma} \vdash \hat{v} : \hat{\tau}_i$ with $\varnothing$. Applying the substitution lemma, $\hat{\Gamma} \vdash [\hat{v}_i/x]\texttt{annot}(e, \varepsilon) : \texttt{annot}(\tau, \varepsilon_s)$ with ε_s. Then $\varepsilon \cup \varepsilon_B = \varepsilon_A = \varepsilon_s$ and $\tau_A = \tau_B = \texttt{annot}(\tau, \varepsilon_s)$.

□

Theorem B.8 (CC Single-step Soundness). *If $\hat{\Gamma} \vdash \hat{e}_A : \hat{\tau}_A$ with ε_A and $\hat{e}_A$ is not a value, then $\hat{e}_A \longrightarrow \hat{e}_B \mid \varepsilon$, where $\hat{\Gamma} \vdash \hat{e}_B : \hat{\tau}_B$ with ε_B and $\hat{\tau}_B <: \hat{\tau}_A$ and $\varepsilon_B \cup \varepsilon \subseteq \varepsilon_A$, for some $\hat{e}_B$, ε, $\hat{\tau}_B$, and ε_B.*

Theorem B.9 (CC Multi-step Soundness). *If $\hat{\Gamma} \vdash \hat{e}_A : \hat{\tau}_A$ with ε_A and $\hat{e}_A \longrightarrow^* e_B \mid \varepsilon$, then $\hat{\Gamma} \vdash \hat{e}_B : \hat{\tau}_B$ with ε_B, where $\hat{\tau}_B <: \hat{\tau}_A$ and $\varepsilon_B \cup \varepsilon \subseteq \varepsilon_A$, for some $\hat{\tau}_B$, ε_B.*

Proof. The same as for OC. □

C DESUGARING

REFERENCES

- [1] Christian Bird, Nachiappan Nagappan, Brendan Murphy, Harald Gall, and Premkumar Devanbu. 2011. Don't Touch My Code!: Examining the Effects of Ownership on Software Quality. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE '11).* ACM, New York, NY, USA, 4–14. DOI: http://dx.doi.org/10.1145/2025113.2025119
- [2] Shuo Chen, David Ross, and Yi-Min Wang. 2007. An Analysis of Browser Domain-isolation Bugs and a Light-weight Transparent Defense Mechanism. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07).* ACM, New York, NY, USA, 2–11. DOI: http://dx.doi.org/10.1145/1315245.1315248
- [3] Zack Coker, Michael Maass, Tianyuan Ding, Claire Le Goues, and Joshua Sunshine. 2015. Evaluating the Flexibility of the Java Sandbox. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC 2015).* ACM, New York, NY, USA, 1–10. DOI: http://dx.doi.org/10.1145/2818000.2818003
- [4] Jack B. Dennis and Earl C. Van Horn. 1966. Programming Semantics for Multiprogrammed Computations. *Commun. ACM* 9, 3 (March 1966), 143–155. DOI: http://dx.doi.org/10.1145/365230.365252
- [5] Raulf Kneuper. 1997. Limits of Formal Methods. *Formal Aspects of Computing* 3, 1 (1997).
- [6] J. M. Lucassen and D. K. Gifford. 1988. Polymorphic Effect Systems. In *Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '88).* ACM, New York, NY, USA, 47–57. DOI: http://dx.doi.org/10.1145/73560.73564
- [7] Michael Maass. 2016. *A Theory and Tools for Applying Sandboxes Effectively.* Ph.D. Dissertation. Carnegie Mellon University.
- [8] Sergio Maffeis, John C. Mitchell, and Ankur Taly. 2010. Object Capabilities and Isolation of Untrusted Web Applications. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP '10).* IEEE Computer Society, Washington, DC, USA, 125–140. DOI: http://dx.doi.org/10.1109/SP.2010.16
- [9] Mark S. Miller. 2006. *Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control.* Ph.D. Dissertation. Johns Hopkins University.
- [10] Flemming Nielson and Hanne Riis Nelson. 1999. Type and Effect Systems. *Commun. ACM* (1999), 114–136. DOI: http://dx.doi.org/10.1145/361604.361612
- [11] Ligia Nistor, Darya Kurilova, Stephanie Balzer, Benjamin Chung, Alex Potanin, and Jonathan Aldrich. 2013. Wyvern: A Simple, Typed, and Pure Object-oriented Language. In *Proceedings of the 5th Workshop on Mechanisms for Specialization, Generalization and inHerItance (MASPEGHI '13).* ACM, New York, NY, USA, 9–16. DOI: http://dx.doi.org/10.1145/2489828.2489830
- [12] Lukas Rytz, Martin Odersky, and Philipp Haller. 2012. Lightweight Polymorphic Effects. In *Proceedings of the 26th European Conference on Object-Oriented Programming (ECOOP'12).* Springer-Verlag, Berlin, Heidelberg, 258–282. DOI: http://dx.doi.org/10.1007/978-3-642-31057-7_13
- [13] Jerome H. Saltzer. 1974. Protection and the Control of Information Sharing in Multics. *Commun. ACM* 17, 7 (1974), 388–402.
- [14] Jerome H. Saltzer and Michael D. Schroeder. 1975. The Protection of Information in Computer Systems. In *Proceedings of the IEEE 63-9.*
- [15] Z. Cliffe Schreuders, Tanya Mcgill, and Christian Payne. 2013. The State of the Art of Application Restrictions and Sandboxes: A Survey of Application-oriented Access Controls and Their Shortfalls. *Computers and Security* 32 (2013), 219–241.
- [16] J.-P. Talpin and P. Jouvelot. 1994. The type and effect discipline. *Information and Computation* 111, 2 (1994), 245–296.
- [17] Y.-M. Tang. 1994. *Control-Flow Analysis by Effect Systems and Abstract Interpretation.* Ph.D. Dissertation. Ecole des Mines de Paris.
- [18] Mike Ter Louw, Prithvi Bisht, and V Venkatakrishnan. 2008. Analysis of Hypertext Isolation Techniques for XSS Prevention. *Web 2.0 Security and Privacy* (2008).
- [19] Robert N. M. Watson. 2007. Exploiting Concurrency Vulnerabilities in System Call Wrappers. In *USENIX Workshop on Offensive Technologies.* Article 2, 8 pages.