A Structural Soundness Proof for Shivers's Escape Technique

A Case for Galois Connections

Jan Midtgaard[1], Michael D. Adams[2], and Matthew Might[3]

[1] Aarhus University, Denmark
[2] Portland State University, USA
[3] University of Utah, USA

Abstract. Shivers's escape technique enables one to analyse the control flow of higher-order program fragments. It is widely used, but its soundness has never been proven. In this paper, we present the first soundness proof for the technique. Our proof is structured as a composition of Galois connections and thus rests on the foundations of abstract interpretation.

1 Introduction

Control-flow analysis is traditionally a *whole program analysis* [Nielson et al., 1999] meaning that it needs access to the entire program text. As flow-analysis algorithms such as 0CFA require cubic time in the size of the program,[1] this limits their applicability to large programs.

Techniques exist, however, for analysing only a part of a program (e.g., an independent module). One such technique is Shivers's escape technique [Shivers, 1991, Sec. 3.8.2]:

> *"Our abstract analysis can handle this by defining two special tokens: the external procedure* xproc*, and the external call* xcall*. The* xproc *represents unknown procedures that are passed into our program from the outside world at run time. The* xcall *represents calls to procedures that happen external to the program text.*
>
> ...
>
> *We maintain a set ESCAPED of escaped procedures, which initially contains* xproc *and the top-level lambda of the program. The rules for the external call, the external procedure and escaped functions are simple:*
>
> 1. *Any procedure passed to the external procedure escapes.*
> 2. *Any escaped procedure can be called from the external call.*
> 3. *When a procedure is called from the external call, it may be applied to any escaped procedure."*

[1] For *typed* programs the complexity is usually not that bad [Heintze and McAllester, 1997].

A. Miné and D. Schmidt (Eds.): SAS 2012, LNCS 7460, pp. 352–369, 2012.

$$
\begin{aligned}
SExp \ni s &::= (t_0\ t_1 \ldots t_n)^{\ell} && \text{(application)}\\
TExp \ni t &::= x^{\ell} && \text{(variable)}\\
&\mid\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell} && \text{(function)}
\end{aligned}
$$

Fig. 1. CPS language

Shivers does not prove his technique to be sound, however. In this paper, we show how his technique can be derived using abstract interpretation by composing a number of well-known Galois connections.

We wish to stress that the escape technique presented in this paper is applicable to any higher-order program analysis even though we present it in terms of a higher-order language in continuation-passing style. It is thus as relevant to a higher-order language like JavaScript as it is to a higher-order language like Scheme. This proof technique grew out of an unpublished soundness proof for the fast type-recovery of Adams et al. [2011].

2 Control-Flow Analysis

To focus on the topic at hand, namely modularity, we limit ourselves to a core language consisting of the lambda calculus in *continuation-passing style* (CPS). The grammar of the language is presented in Figure 1. Following Reynolds [1998] the grammar distinguishes *serious* expressions (*SExp*) whose evaluation may diverge from *trivial* expressions (*TExp*) whose evaluation is guaranteed to terminate. As is standard [Nielson et al., 1999], we label all sub-expressions with a unique label ℓ to distinguish different occurrences of the same sub-expression. For the remainder of this paper, we let labels on variables be implicit to ease the syntactic overhead.

There are a number of advantages to the small-step CPS framework. First, since all intermediate results are bound to a variable, an analysis can be characterized in terms of computing an abstract environment or store. One would otherwise need to compute an *abstract cache* that maps labels to abstract values [Nielson et al., 1999]. Second, since all calls are *tail calls*, the analysis does not need special measures to propagate *return flow*. This is instead handled by bindings to continuation variables. CPS therefore makes for a simple, uniform analysis.

The control-flow analysis is formulated in terms of the curried transfer function T defined in Figure 2. For a given program P, the analysis is defined as the least fixed point of $T(P)$. The analysis computes an abstract environment, $\rho : \mathit{Var} \to \mathit{Val}$, which approximates the bindings of an actual program run. T relies on a helper function E for analysing trivial expressions. We furthermore use the shorthand notation $[\overline{x} \mapsto E(\overline{t}, \rho)]$ to mean $[x_1 \mapsto E(t_1, \rho), \ldots, x_n \mapsto E(t_n, \rho)]$. T considers all call sites $(t_0\ t_1 \ldots t_n)^{\ell}$ of the program P in each iteration. This is easily accomplished by a traversal of P's abstract syntax tree. Here we simply

$$T : \wp(SExp) \to (Var \to Val) \to (Var \to Val)$$

$$T(P)(\rho) = \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in P \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0,\rho)}} \rho \sqcup [\overline{x} \mapsto E(\overline{t}, \rho)]$$

where

$$E(x, \rho) = \rho(x)$$
$$E((\lambda\, x_1 \ldots x_n.\ s)^{\ell}, \rho) = \{(\lambda\, x_1 \ldots x_n.\ s)^{\ell}\}$$

Fig. 2. CPS analysis

$$Var = XVar + IVar \quad \text{(variables)}$$
$$Lam = XLam + ILam \quad \text{(functions)}$$
$$Val = \wp(Lam) \quad \text{(values)}$$

$$TExp = XTExp + ITExp \quad \text{(trivial exprs)}$$
$$SExp = XSexp + ISexp \quad \text{(serious exprs)}$$

Fig. 3. Syntactic and analysis domains

express P in terms of a set of call sites. For each possible receiver of a call, the analysis binds the (analysis result of the) actual parameters to the formals. This analysis agrees with the 0CFAs of Midtgaard and Jensen [2008] and Might [2010] (sans reachability) and is therefore known to be sound.

We define the domains for the refined analysis in Figure 3. To pave the way for a CFA over open programs, we split the domains into disjoint *external* and *internal* sets and assume some basic consistencies among them. Variables bound in an internal lambda are all internal variables. An analogous constraint applies to external variables and external lambdas. Similarly, trivial sub-expressions of an internal serious expression are all internal trivial expressions. However, the trivial sub-expressions of an external serious expression may be either internal or external.

For example, consider an analysis restricted to the boxed expression below. The sub-expressions outside the box are external while those inside the box are internal. Note that, inside the box, the variable occurrence of k is an internal expression but refers to the external variable k.

$$(\lambda\, k.\ (k\ \boxed{(\lambda\, x.\ (k\ x))}))$$

Finally, we assume that internal variables must be located inside an internal lambda. Hence, for an external call site $(t_0\ t_1 \ldots t_n)$ none of the t_j can be internal variables. If t_j is an internal lambda located immediately inside such an external call site, we include it in a dedicated set $Toplevel \subset ILam$.

3 Abstract Interpretation

A Galois connection is a pair of functions (the *adjoints*) $\alpha : C \to A$ and $\gamma : A \to C$ which connect two partially ordered sets $\langle C; \sqsubseteq \rangle$ and $\langle A; \leq \rangle$ such that:

$$\forall c \in C, a \in A \ : \ \alpha(c) \leq a \iff c \sqsubseteq \gamma(a)$$

Following abstract interpretation tradition [Cousot and Cousot, 1994], we typeset Galois connections as $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle$.

Galois connections enjoy a number of properties. First, α and γ are necessarily monotone. Second, the composition $\gamma \circ \alpha$ is extensive ($\forall c \in C : c \sqsubseteq \gamma \circ \alpha(c)$) and the composition $\alpha \circ \gamma$ is reductive ($\forall a \in A : \alpha \circ \gamma(a) \leq a$). For Galois connections with a surjective α (or equivalently with an injective γ), the latter composition yields the identity $\alpha \circ \gamma = 1$. These are called Galois surjections (or Galois insertions) and are typeset as $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows\!\!\!\!\!\rightarrow}} \langle A; \leq \rangle$. When both α and γ are surjective, the Galois connection is an isomorphism and is typeset as $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftarrow\!\!\!\!\!\rightarrow}} \langle A; \leq \rangle$.

Galois connections that connect complete lattices have even more properties. For example, α is a complete join morphism (CJM) and thus preserves joins (i.e., $\alpha(\sqcup_i S_i) = \vee_i \alpha(S_i)$), and γ is a complete meet morphism and thus preserves meets (i.e., $\gamma(\wedge_i S_i) = \sqcap_i \gamma(S_i)$). For easy reference, we summarize in Figure 4 the Galois connections relevant to this paper. Following Might [2010] we typeset them as inference rules. For the purposes of this paper they all connect complete lattices.

Galois connections interact nicely with fixed points. Given a Galois connection between complete lattices and a monotone function F, the *fixed-point transfer theorem* [Cousot and Cousot, 1979] provides an approximation of $\operatorname{lfp} F$:

$$\alpha(\operatorname{lfp} F) \leq \operatorname{lfp}(\alpha \circ F \circ \gamma) \leq \operatorname{lfp} F^\sharp$$

Here, $F^\sharp$ is a monotone function such that $\alpha \circ F \circ \gamma \dot{\leq} F^\sharp$. Whereas any $F^\sharp$ satisfying these requirements will do, the *best abstraction* satisfying $F^\sharp = \alpha \circ F \circ \gamma$ represents the best possible function over the chosen abstract domain [Cousot and Cousot, 1992]. In the calculational approach to abstract interpretation, Cousot [1999] advocates simple algebraic manipulation to find such a function (if it exists) or a sound approximation thereof.

When F expresses an execution step in the formal semantics for a program, $\operatorname{lfp} F$ describes the *collecting semantics* of the program: an ideal but generally uncomputable exploration of program paths that is subject to over approximation.

4 Abstracting the Domains

We derive Shivers's escape technique in two steps. In this section, we define Galois connections that abstract over the domains of our analysis. Then, in

Transitive abstraction [Cousot and Cousot, 1994]

$$\frac{\langle D_0; \sqsubseteq_0\rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle D_1; \sqsubseteq_1\rangle \qquad \langle D_1; \sqsubseteq_1\rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle D_2; \sqsubseteq_2\rangle}{\langle D_0; \sqsubseteq_0\rangle \underset{\alpha_2 \circ \alpha_1}{\overset{\gamma_1 \circ \gamma_2}{\leftrightarrows}} \langle D_2; \sqsubseteq_2\rangle} \textsc{Trans}$$

Elementwise abstraction [Cousot and Cousot, 1997]

$$\frac{@ : C \to A}{\langle \wp(C); \subseteq\rangle \underset{\alpha_@ = \lambda P.\, \{@(p) \mid p \in P\}}{\overset{\gamma_@ = \lambda Q.\, \{p \mid @(p) \in Q\}}{\twoheadleftarrow\!\!\!\!\twoheadrightarrow}} \langle \wp(A); \subseteq\rangle} \textsc{Element}$$

Isomorphic maps

$$\frac{}{\langle (A+B) \to C; \dot{\sqsubseteq}\rangle \underset{\alpha_\sim = \lambda f.\, (f|_A, f|_B)}{\overset{\gamma_\sim = \lambda(g,h).\, \lambda x. \begin{cases} g(x) & x \in A \\ h(x) & x \in B \end{cases}}{\twoheadleftarrow\!\!\!\!\twoheadrightarrow}} \langle (A \to C) \times (B \to C); \dot{\sqsubseteq} \times \dot{\sqsubseteq}\rangle} \textsc{Iso}$$

Collapsing abstraction

$$\frac{}{\langle D \to \wp(C); \dot{\subseteq}\rangle \underset{\alpha_\cup = \lambda f.\, \cup_{x \in Dom(f)} f(x)}{\overset{\gamma_\cup = \lambda s.\, \lambda x.\, s}{\twoheadleftarrow\!\!\!\!\twoheadrightarrow}} \langle \wp(C); \subseteq\rangle} \textsc{Collapse}$$

Pointwise abstraction [Cousot and Cousot, 1994]

$$\frac{\langle \wp(C); \subseteq\rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle A; \sqsubseteq\rangle}{\langle D \to \wp(C); \dot{\subseteq}\rangle \underset{\alpha_. = \lambda f.\, \lambda x.\, \alpha_1(f(x))}{\overset{\gamma_. = \lambda f.\, \lambda x.\, \gamma_1(f(x))}{\leftrightarrows}} \langle D \to A; \dot{\sqsubseteq}\rangle} \textsc{Pointwise}$$

Product abstraction [Cousot and Cousot, 1994]

$$\frac{\langle C_1; \sqsubseteq_1\rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle A_1; \leq_1\rangle \qquad \langle C_2; \sqsubseteq_2\rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle A_2; \leq_2\rangle}{\langle C_1 \times C_2; \sqsubseteq_1 \times \sqsubseteq_2\rangle \underset{\alpha_\times = \lambda(c_1,c_2).\, (\alpha_1(c_1), \alpha_2(c_2))}{\overset{\gamma_\times = \lambda(a_1,a_2).\, (\gamma_1(a_1), \gamma_2(a_2))}{\leftrightarrows}} \langle A_1 \times A_2; \leq_1 \times \leq_2\rangle} \textsc{Component}$$

Subset abstraction [Cousot and Cousot, 1997]

$$\frac{S \subset C}{\langle \wp(C); \subseteq\rangle \underset{\alpha_\subset = \lambda c.\, c \cap S}{\overset{\gamma_\subset = \lambda s.\, s \cup (C \setminus S)}{\twoheadleftarrow\!\!\!\!\twoheadrightarrow}} \langle \wp(S); \subseteq\rangle} \textsc{Subset}$$

Fig. 4. Galois connection reference

Section 5, we use these abstractions to derive the transfer function of an analysis incorporating Shivers's escape technique.

Figure 5 provides an overview of the Galois connections defined in this section using the judgments defined in Figure 4.

4.1 Abstracting Values

The operator @ : $Lam \to ILam + \{\mathbf{xproc}\}$ maps lambdas to either internal lambdas or the dedicated token **xproc** representing all external procedures:

$$@((\lambda\, x_1 \ldots x_n.\ s)^\ell) = \begin{cases} (\lambda\, x_1 \ldots x_n.\ s)^\ell & (\lambda\, x_1 \ldots x_n.\ s)^\ell \in ILam \\ \mathbf{xproc} & (\lambda\, x_1 \ldots x_n.\ s)^\ell \in XLam \end{cases}$$

Using @, both Element judgments in Figure 5 build an elementwise abstraction on values. Since @ is surjective, the resulting Galois connection is a Galois surjection:

$$\wp(Lam) \underset{\alpha_@}{\overset{\gamma_@}{\leftleftarrows}} \wp(ILam + \{\mathbf{xproc}\})$$

4.2 Abstracting the Store

We abstract the store by, first, mapping the store to an isomorphic representation containing two stores: one for external bindings and one for internal bindings. Then, we abstract each component individually. By transitivity, the resulting abstraction is a Galois connection.

The Iso judgment in Figure 5 uses the fact that $Var = XVar + IVar$ and an isomorphic representation of the store to build the following Galois connection:

$$(XVar + IVar) \to Val \underset{\alpha_\sim}{\overset{\gamma_\sim}{\leftleftarrows}} (XVar \to Val) \times (IVar \to Val)$$

This isomorphism is well-known within set theory [Winskel, 2010], semantics, and functional programming [Wand and Vaillancourt, 2004]. It allows us to abstract the external bindings separately from the internal bindings.

Next, the Collapse judgment in Figure 5 abstracts the external bindings with a collapsing abstraction $\alpha_\cup$ that join all bindings and aliases them into a single set of values:

$$XVar \to \wp(Lam) \underset{\alpha_\cup}{\overset{\gamma_\cup}{\leftleftarrows}} \wp(Lam)$$

The Pointwise judgment in Figure 5 abstracts the internal bindings using a standard pointwise lifting of the value abstraction:

$$IVar \to \wp(Lam) \underset{\alpha_\cdot}{\overset{\gamma_\cdot}{\leftleftarrows}} IVar \to \wp(ILam + \{\mathbf{xproc}\})$$

Finally, the Component judgment composes the two abstractions to form a product abstraction of both external and internal bindings:

$$(XVar \to Val) \times (IVar \to Val) \underset{\alpha_\times}{\overset{\gamma_\times}{\leftleftarrows}} \wp(ILam + \{\mathbf{xproc}\}) \times (IVar \to \wp(ILam + \{\mathbf{xproc}\}))$$

$$\dfrac{\dfrac{}{XVar \to Val \leftrightarrows \wp(Lam)}\textsc{Collapse} \qquad \dfrac{}{\wp(Lam) \leftrightarrows\!\!\!\!\rightarrow \wp(ILam + \{\mathbf{xproc}\})}\textsc{Element}}{XVar \to Val \leftrightarrows \wp(ILam + \{\mathbf{xproc}\})}\textsc{Trans}\ (1)$$

$$\dfrac{\dfrac{}{Var \to Val \leftarrow\!\!\!\!\leftrightarrows (XVar \to Val) \times (IVar \to Val)}\textsc{Iso} \qquad \dfrac{\dfrac{(1)}{XVar \to Val \leftrightarrows \wp(ILam + \{\mathbf{xproc}\})} \qquad \dfrac{\dfrac{}{Val \leftrightarrows\!\!\!\!\rightarrow \wp(ILam + \{\mathbf{xproc}\})}\textsc{Element}}{IVar \to Val \leftrightarrows IVar \to \wp(ILam + \{\mathbf{xproc}\})}\textsc{Pointwise}}{(XVar \to Val) \times (IVar \to Val) \leftrightarrows \wp(ILam + \{\mathbf{xproc}\}) \times (IVar \to \wp(ILam + \{\mathbf{xproc}\}))}\textsc{Component}}{Var \to Val \leftrightarrows \wp(ILam + \{\mathbf{xproc}\}) \times (IVar \to \wp(ILam + \{\mathbf{xproc}\}))}\textsc{Trans}$$

Fig. 5. Galois connection inference tree

4.3 Abstracting Programs

The analysis in Figure 2 computes a join for each call site of the input program P. When only a part of the program is available, we represent the information loss as an abstraction of the set of call sites. This is formulated as a subset abstraction where P omits *XSexp* and keeps only *ISexp*:

$$\wp(SExp) \underset{\alpha_\subset}{\overset{\gamma_\subset}{\leftrightarrows}} \wp(ISexp)$$

5 Abstracting the Analysis

In this section, we use the Galois connections defined in Section 4 to abstract T and derive a new transfer function, $T^\sharp$, that is sound with respect to T. By the fixed-point transfer theorem [Cousot and Cousot, 1979], the fixed point of $T^\sharp$ is a sound approximation of the fixed point of T.

5.1 Abstracting the Helper Function

We calculate a sound approximation of E, the helper function defined in Figure 2, by composing it with the adjoints of the Galois connections.

$$\begin{aligned}
&\alpha_@ \circ E(t, \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i)) && \text{(def. of } E\text{)}\\
&= \begin{cases} \alpha_@((\gamma_\sim \circ \gamma_\times(\rho_e, \rho_i))(x)) & t = x \\ \alpha_@(\{(\lambda\, x_1 \ldots x_n.\ s)^\ell\}) & t = (\lambda\, x_1 \ldots x_n.\ s)^\ell \end{cases} && \text{(def. of } \gamma_\times\text{)}\\
&= \begin{cases} \alpha_@((\gamma_\sim(\gamma_\cup \circ \gamma_@(\rho_e), \gamma_\cdot(\rho_i)))(x)) & t = x \\ \alpha_@(\{(\lambda\, x_1 \ldots x_n.\ s)^\ell\}) & t = (\lambda\, x_1 \ldots x_n.\ s)^\ell \end{cases} && \text{(def. of } \gamma_\sim\text{)}\\
&= \begin{cases} \alpha_@((\gamma_\cup \circ \gamma_@(\rho_e))(x)) & t = x \in XVar \\ \alpha_@((\gamma_\cdot(\rho_i))(x)) & t = x \in IVar \\ \alpha_@(\{(\lambda\, x_1 \ldots x_n.\ s)^\ell\}) & t = (\lambda\, x_1 \ldots x_n.\ s)^\ell \end{cases} && \text{(def. of } \gamma_\cup\text{)}\\
&= \begin{cases} \alpha_@(\gamma_@(\rho_e)) & t = x \in XVar \\ \alpha_@((\gamma_\cdot(\rho_i))(x)) & t = x \in IVar \\ \alpha_@(\{(\lambda\, x_1 \ldots x_n.\ s)^\ell\}) & t = (\lambda\, x_1 \ldots x_n.\ s)^\ell \end{cases} && \text{(def. of } \gamma_\cdot\text{)}\\
&= \begin{cases} \alpha_@(\gamma_@(\rho_e)) & t = x \in XVar \\ \alpha_@(\gamma_@(\rho_i(x))) & t = x \in IVar \\ \alpha_@(\{(\lambda\, x_1 \ldots x_n.\ s)^\ell\}) & t = (\lambda\, x_1 \ldots x_n.\ s)^\ell \end{cases} && \text{(Galois surjection)}\\
&= \begin{cases} \rho_e & t = x \in XVar \\ \rho_i(x) & t = x \in IVar \\ \alpha_@(\{(\lambda\, x_1 \ldots x_n.\ s)^\ell\}) & t = (\lambda\, x_1 \ldots x_n.\ s)^\ell \end{cases} && \text{(def. of } \alpha_@\text{)}\\
&= \begin{cases} \rho_e & t = x \in XVar \\ \rho_i(x) & t = x \in IVar \\ \{\mathbf{xproc}\} & t = (\lambda\, x_1 \ldots x_n.\ s)^\ell \in XLam \\ \{(\lambda\, x_1 \ldots x_n.\ s)^\ell\} & t = (\lambda\, x_1 \ldots x_n.\ s)^\ell \in ILam \end{cases}
\end{aligned}$$

Hence by defining $\widehat{E}$ as:

$$\widehat{E}(t,\rho_e,\rho_i) = \begin{cases} \rho_e & t = x \in \mathit{XVar} \\ \rho_i(x) & t = x \in \mathit{IVar} \\ \{\mathbf{xproc}\} & t = (\lambda\, x_1 \ldots x_n.\ s)^{\ell} \in \mathit{XLam} \\ \{(\lambda\, x_1 \ldots x_n.\ s)^{\ell}\} & t = (\lambda\, x_1 \ldots x_n.\ s)^{\ell} \in \mathit{ILam} \end{cases}$$

the following lemma holds by construction.

Lemma 1 ($\widehat{E}$ is the best abstraction of E)

$$\forall t, \rho_e, \rho_i \ :\ \alpha_{@} \circ E(t, \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i)) = \widehat{E}(t, \rho_e, \rho_i)$$

While $\widehat{E}$ is not an operator from a domain to itself, it nevertheless represents the best abstraction of the operator E in terms of the abstract arguments ρ_e and ρ_i.
By inspecting $\widehat{E}$ applied to external expressions, we have the following bound.

Lemma 2 (Upper bound on $\widehat{E}$)

$$\forall t \in \mathit{XTExp}, \rho_e, \rho_i \ :\ \widehat{E}(t, \rho_e, \rho_i) \subseteq \rho_e \cup \{\mathbf{xproc}\}$$

By a simple case analysis on t, we furthermore discover that $\widehat{E}$ is monotone in its environment arguments, ρ_e and ρ_i.

Lemma 3 ($\widehat{E}$ is monotone in environment arguments)

$$\forall t, \rho_e, \rho'_e, \rho_i, \rho'_i \ :\ (\rho_e, \rho_i) \sqsubseteq (\rho'_e, \rho'_i) \implies \widehat{E}(t, \rho_e, \rho_i) \subseteq \widehat{E}(t, \rho'_e, \rho'_i)$$

5.2 Abstracting the Transfer Function

We now construct the abstract transfer function $T^{\sharp}$ by composing T with the adjoints of the Galois connections. Given P_i, ρ_e, and ρ_i, we have:

$$\begin{aligned}
& \alpha_{\times} \circ \alpha_{\sim} \circ (T(\gamma_{\subset}(P_i))) \circ \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i) \\
= \ & \ldots \\
\sqsubseteq \ & (\rho_e \cup \{\mathbf{xproc}\} \cup \mathit{Toplevel}, \rho_i) \\
& \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq P_i \\ \mathbf{xproc} \in \widehat{E}(t_0, \rho_e, \rho_i)}} (\rho_e \cup \bigcup_{j \in [1;n]} \widehat{E}(t_j, \rho_e, \rho_i), \rho_i) \\
& \sqcup \bigsqcup_{(\lambda\, x_1 \ldots x_n.\ s)^{\ell} \in \rho_e \cup \mathit{Toplevel}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto (\rho_e \cup \{\mathbf{xproc}\} \cup \mathit{Toplevel})]) \\
& \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq P_i \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0, \rho_e, \rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t}, \rho_e, \rho_i)])
\end{aligned}$$

The full calculation is lengthy and is therefore deferred to Appendix A. Nonetheless, it proceeds from simple algebraic rewritings relying only on standard Galois-connection reasoning.

By defining the abstract transfer function $T^\sharp$ as:

$$T^\sharp : \wp(\mathit{ISexp}) \to \widehat{\mathit{Env}} \to \widehat{\mathit{Env}}$$

$$\begin{aligned} T^\sharp(P_i)(\rho_e,\rho_i) = (\rho_e \cup \{\mathbf{xproc}\} \cup \mathit{Toplevel}, \rho_i) \\ \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^\ell\} \subseteq P_i \\ \mathbf{xproc} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e \cup \bigcup_{j\in[1;n]} \widehat{E}(t_j,\rho_e,\rho_i), \rho_i) \\ \sqcup \bigsqcup_{(\lambda\, x_1 \ldots x_n.\ s)^\ell \in \rho_e \cup \mathit{Toplevel}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto (\rho_e \cup \{\mathbf{xproc}\} \cup \mathit{Toplevel})]) \\ \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^\ell\} \subseteq P_i \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t},\rho_e,\rho_i)]) \end{aligned}$$

where $\widehat{\mathit{Env}} = \wp(\mathit{ILam} + \{\mathbf{xproc}\}) \times (\mathit{IVar} \to \wp(\mathit{ILam} + \{\mathbf{xproc}\}))$
the following lemma holds by construction.

Lemma 4 ($T^\sharp$ is a sound approximation of T)

$$\forall P_i, \rho_e, \rho_i \ : \ \alpha_\times \circ \alpha_\sim \circ (T(\gamma_\subset(P_i))) \circ \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i) \sqsubseteq T^\sharp(P_i)(\rho_e,\rho_i)$$

By a sequence of upward judgments ($\sqsubseteq$) from ρ_e to ρ'_e, and from ρ_i to ρ'_i and by appeal to Lemma 3 we can furthermore verify that the derived transfer function is monotone.

Lemma 5 ($T^\sharp$ is monotone)

$$\forall P_i, \rho_e, \rho'_e, \rho_i, \rho'_i \ : \ (\rho_e,\rho_i) \sqsubseteq (\rho'_e,\rho'_i) \implies T^\sharp(P_i)(\rho_e,\rho_i) \sqsubseteq T^\sharp(P_i)(\rho'_e,\rho'_i)$$

Finally, the soundness of the derived analysis follows from the fixed-point transfer theorem [Cousot and Cousot, 1979]:

Theorem 1 (Soundness of the analysis with Shivers's escape technique)

$$\forall P_i \ : \ \alpha_\times \circ \alpha_\sim(\mathrm{lfp}\, T(\gamma_\subset(P_i))) \sqsubseteq \mathrm{lfp}\, T^\sharp(P_i)$$

5.3 Proof Summary

The soundness of the analysis (Theorem 1) is proven using the fixed-point transfer theorem. In order to use the fixed-point transfer theorem, we construct a Galois connection between the domains of T and $T^\sharp$ (Section 4), prove that $T^\sharp$ is a sound approximation of T (Lemma 4) and prove that $T^\sharp$ is monotone (Lemma 5).

Since T includes a helper function, E, we also abstract E to $E^\sharp$. Lemmas 1 and 2 simplify the calculations relating to $E^\sharp$ in the proof of Lemma 4. We use the fact that $E^\sharp$ is monotone (Lemma 3) in the proof that $T^\sharp$ is monotone (Lemma 5).

$$\textsc{XProc}\ \frac{}{(\{\mathbf{xproc}\} \cup \mathit{Toplevel}) \subseteq \rho_e} \qquad \frac{(t_0\ t_1 \ldots t_n)^{\ell} \in P_i \qquad \mathbf{xproc} \in \widehat{E}(t_0, \rho_e, \rho_i)}{\widehat{E}(t_j, \rho_e, \rho_i) \subseteq \rho_e, \qquad j \in [1; n]}\ \textsc{Escape}$$

$$\textsc{XCall}\ \frac{(\lambda\, x_1 \ldots x_n.\ s)^{\ell} \in (\rho_e \cup \mathit{Toplevel})}{(\rho_e \cup \{\mathbf{xproc}\} \cup \mathit{Toplevel}) \subseteq \rho_i(\overline{x})}$$

$$\frac{(t_0\ t_1 \ldots t_n)^{\ell} \in P_i \qquad (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0, \rho_e, \rho_i)}{\widehat{E}(\overline{t}, \rho_e, \rho_i) \subseteq \rho_i(\overline{x})}\ \textsc{ICall}$$

Fig. 6. CFA constraints

6 Extracting Constraints

Given the transfer function $T^\sharp$, we are now in a position to take a step backwards and extract constraints equivalent to $T^\sharp$ [Cousot and Cousot, 1995]. For any post-fixed point (ρ_e, ρ_i) of $T^\sharp$, it holds that $T^\sharp(P_i)(\rho_e, \rho_i) \sqsubseteq (\rho_e, \rho_i)$. This is equivalent to the constraint rules in Figure 6.

The XCall constraint is needlessly complex, however, as XProc guarantees that both $\{\mathbf{xproc}\}$ and *Toplevel* are already subsets of ρ_e. Hence we can simplify XCall to:

$$\textsc{XCall'}\ \frac{(\lambda\, x_1 \ldots x_n.\ s)^{\ell} \in \rho_e}{\rho_e \subseteq \rho_i(\overline{x})}$$

The resulting constraints can be understood as follows.

- XProc: External procedures and top-level procedures may escape.
- Escape: If a call-site may target an external procedure, all of the actual parameters escape.
- XCall': If a procedure escapes, then its formal parameters may take any escaped value.
- ICall: For internal call-sites and procedures, values flow from the actual parameters to the formal parameters (as in the base analysis).

It is striking how close these constraints are to Shivers's original description as quoted in Section 1. In our characterization, the external environment ρ_e plays the role of Shivers's ESCAPED set. The two descriptions differ in that we have not found the need to abstract external call-sites into a dedicated **xcall** token. Doing so can be achieved by replacing the subset abstraction by another elementwise abstraction over call sites. In his description, Shivers also omits the detail that external (free) variables should be looked up in ESCAPED (i.e., ρ_e).

An implementation of the analysis can be realized as a direct implementation of the transfer function $T^\sharp$ by performing Kleene iteration or by outputting *conditional constraints* based on Figure 6 in the style of Palsberg and Schwartzbach [1995] and subsequently solving them in $O(n^3)$ time.

7 Related Work

This work derives from the Galois-connection school of abstract interpretation [Cousot and Cousot, 1979]. Previous work by the present authors investigate derivations of CFAs using Galois connections [Midtgaard and Jensen, 2008, 2012; Might, 2010].

Shivers [1991, Sec. 3.8.2] conceived of the escaping-lambdas technique using **xproc** to denote an external procedure, **xcall** to denote an external call, and ESCAPED to denote the set of escaping procedures. However, he did not prove the soundness of the technique. Serrano and Feeley [1996] used a similar concept of escaping to the top of the lattice in their development of modular analyses for both first-order and higher-order languages. Ashley and Dybvig [1998] later used the escaping-to-top idea to formulate a sub-cubic CFA by jumping to top if more than a constant number of procedures flow to a particular variable. The implementation described in Ashley's dissertation [Ashley, 1996, Sec. 6.1.1] furthermore uses an escape set to accommodate free variables. However, Ashley's soundness proof assumes programs are closed. The present authors [Adams et al., 2011] have recently combined the escaping-to-top idea with novel algorithms and data structures to develop a fast, flow-sensitive type-recovery analysis. We did not prove soundness of the escape technique in that work.

Flanagan and Felleisen [1999] developed a *componential* set-based analysis. Their approach extends the set-based analysis by Heintze [1992] by avoiding re-extracting constraints from unmodified program modules upon later re-analysis. As a consequence, they achieve substantial speed-ups in their interactive setting of a static debugger [Flanagan, 1997]. In a follow-up paper, Meunier et al. [2006] develop a set-based analysis for program modules with *contracts*. The contracts enable their analysis to statically detect and pin-point possible breaches (i.e., "blame" in the terminology of the contract literature).

Lee et al. [2002] construct 0CFA/m, a 0CFA variant extended to modules, which analyses a program's modules in order of dependence. The precision of their 0CFA/m is better than a standard 0CFA as it avoids some of the spurious flows of a standard 0CFA. In an accompanying technical report, they prove it sound with respect to *module-variant 0CFA*, an instantiation of Nielson and Nielson's infinitary collecting semantics [Nielson and Nielson, 1997]. Whereas the overall goal of our work agrees with that of Lee et al. [2002], it differs in that our reconstruction of Shivers's escape technique is a sound approximation of the base analysis, 0CFA. As such, it is still monovariant, whereas 0CFA/m is not.

The present paper and the above work focus on untyped programs, but others have investigated modular CFA for typed programs. Banerjee and Jensen [2003] developed a modular and polyvariant CFA based on intersection types for simply-typed programs with recursive function definitions. Like Shivers's untyped escape technique, it handles sub-expressions with free variables. Banerjee and Jensen's analysis is furthermore *compositional* in that the analysis of an expression can be calculated by combining the analysis results of its sub-expressions without re-analysing any of them. Reppy [2006] uses ML's type abstraction to improve the precision of a flow analysis by approximating the arguments of an

abstract type with results computed earlier for the same abstract type. For a broader survey of CFA, we refer the reader to Midtgaard [2012].

Cousot and Cousot [2002] present four strategies for modular program analysis to debunk the myth that abstract interpretation is inherently a whole-program analysis technique. One of these is a *worst-case separate analysis*, which analyses external objects based on no information (i.e, $\top$ in the lattice). Shivers's escape technique goes beyond that approach, by keeping track of previously escaped procedures in the ESCAPED set.

8 Conclusion

Both abstract interpretation and (untyped) control-flow analysis are often presented as inherently whole-program analyses. By characterizing Shivers's CFA escape technique in terms of Galois connections, we show how to extend these to open programs. In doing so, we systematically derive an analysis which is provably sound by construction. Our soundness proof is modular in that the abstraction is structured as a combination of Galois connections. It is furthermore economical in that these Galois connections are well known from the literature. The structure of our approach indicates that staged proofs are a viable way forward for future higher-order analyses. After a base analysis is defined and proven sound, the escape technique can be added and the combination proven sound.

Whereas CPS allows us to focus on the task at hand, one can imagine a number of extensions. For one, our base CFA does not track the reachability of the individual serious expressions. Instead, it conservatively assumes that all sub-expressions are reachable. Adding an additional set to track reachability in the style of Midtgaard and Jensen [2008] and performing a subset abstraction thereof is straightforward. Another extension is to abstract external call-sites to **xcall** as outlined in Section 6 to pave the way for a modular kCFA soundness proof. In such a setting the modularized contours would consist of mixed strings of internal call sites and **xcall** tokens. Characterizing the flat-lattice sub-0CFA [Ashley and Dybvig, 1998] as an abstract interpretation and subsequently its open program extension would be another interesting endeavor.

Acknowledgement. We thank Peter A. Jonsson for comments on an earlier version of this paper.

References

Adams, M.D., Keep, A.W., Midtgaard, J., Might, M., Chauhan, A., Dybvig, R.K.: Flow-sensitive type recovery in linear-log time. In: Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2011), Portland, Oregon (October 2011)

Ashley, J.M.: Flexible and Practical Flow Analysis for Higher-Order Programming Languages. PhD thesis, Department of Computer Science, Indiana University, Bloomington, Indiana (May 1996)

Ashley, J.M., Dybvig, R.K.: A practical and flexible flow analysis for higher-order languages. ACM Transactions on Programming Languages and Systems 20(4), 845–868 (1998)

Banerjee, A., Jensen, T.: Modular control-flow analysis with rank 2 intersection types. Mathematical Structures in Computer Science 13(1), 87–124 (2003)

Cousot, P.: The calculational design of a generic abstract interpreter. In: Broy, M., Steinbrüggen, R. (eds.) Calculational System Design. NATO ASI Series. IOS Press, Amsterdam (1999)

Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: Rosen, B.K. (ed.) Proc. of the Sixth Annual ACM Symposium on Principles of Programming Languages, San Antonio, Texas, pp. 269–282 (January 1979)

Cousot, P., Cousot, R.: Abstract interpretation and application to logic programs. Journal of Logic Programming 13(2-3), 103–179 (1992)

Cousot, P., Cousot, R.: Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages). In: Bal, H. (ed.) Proc. of the Fifth IEEE International Conference on Computer Languages, Toulouse, France, pp. 95–112 (May 1994) (invited paper)

Cousot, P., Cousot, R.: Compositional and Inductive Semantic Definitions in Fixpoint, Equational, Constraint, Closure-condition, Rule-based and Game-Theoretic Form (Invited Paper). In: Wolper, P. (ed.) CAV 1995. LNCS, vol. 939, pp. 293–308. Springer, Heidelberg (1995)

Cousot, P., Cousot, R.: Abstract Interpretation of Algebraic Polynomial Systems. In: Johnson, M. (ed.) AMAST 1997. LNCS, vol. 1349, pp. 138–154. Springer, Heidelberg (1997)

Cousot, P., Cousot, R.: Modular Static Program Analysis. In: Horspool, R.N. (ed.) CC 2002. LNCS, vol. 2304, pp. 159–179. Springer, Heidelberg (2002)

Flanagan, C.: Effective Static Debugging via Componential Set-Based Analysis. PhD thesis, Rice University, Houston, Texas (May 1997)

Flanagan, C., Felleisen, M.: Componential set-based analysis. ACM Transactions on Programming Languages and Systems 21(2), 370–416 (1999)

Heintze, N.: Set-Based Program Analysis. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania (October 1992)

Heintze, N., McAllester, D.: Linear-time subtransitive control flow analysis. In: Cytron, R.K. (ed.) Proc. of the ACM SIGPLAN 1997 Conference on Programming Languages Design and Implementation, Las Vegas, Nevada, pp. 261–272 (June 1997)

Lee, O., Yi, K., Paek, Y.: A proof method for the correctness of modularized 0CFA. Information Processing Letters 81(4), 179–185 (2002)

Meunier, P., Findler, R.B., Felleisen, M.: Modular set-based analysis from contracts. In: Peyton Jones, S. (ed.) Proc. of the 33rd Annual ACM Symposium on Principles of Programming Languages, Charleston, South Carolina, pp. 218–231 (January 2006)

Midtgaard, J.: Control-flow analysis of functional programs. ACM Computing Surveys 44(3) (2012)

Midtgaard, J., Jensen, T.: A Calculational Approach to Control-Flow Analysis by Abstract Interpretation. In: Alpuente, M., Vidal, G. (eds.) SAS 2008. LNCS, vol. 5079, pp. 347–362. Springer, Heidelberg (2008)

Midtgaard, J., Jensen, T.P.: Control-flow analysis of function calls and returns by abstract interpretation. Information and Computation 211, 49–76 (2012); a preliminary version was presented at the 2009 ACM SIGPLAN International Conference on Functional Programming (ICFP 2009)

Might, M.: Abstract Interpreters for Free. In: Cousot, R., Martel, M. (eds.) SAS 2010. LNCS, vol. 6337, pp. 407–421. Springer, Heidelberg (2010)

Nielson, F., Nielson, H.R.: Infinitary control flow analysis: a collecting semantics for closure analysis. In: Jones, N.D. (ed.) Proc. of the 24th Annual ACM Symposium on Principles of Programming Languages, Paris, France, pp. 332–345 (January 1997)

Nielson, F., Nielson, H.R., Hankin, C.: Principles of Program Analysis. Springer (1999)

Palsberg, J., Schwartzbach, M.I.: Safety analysis versus type inference. Information and Computation 118(1), 128–141 (1995)

Reppy, J.: Type-sensitive control-flow analysis. In: Kennedy, A., Pottier, F. (eds.) ML 2006: Proc. of the ACM SIGPLAN 2006 Workshop on ML, pp. 74–83 (September 2006)

Reynolds, J.C.: Definitional interpreters for higher-order programming languages. Higher-Order and Symbolic Computation 11(4), 363–397 (1998); reprinted from the proceedings of the 25th ACM National Conference (1972)

Serrano, M., Feeley, M.: Storage use analysis and its applications. In: Dybvig, R.K. (ed.) Proc. of the First ACM SIGPLAN International Conference on Functional Programming, Philadelphia, Pennsylvania, pp. 50–61 (May 1996)

Shivers, O.: Control-Flow Analysis of Higher-Order Languages or Taming Lambda. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU-CS-91-145 (May 1991)

Wand, M., Vaillancourt, D.: Relating models of backtracking. In: Fisher, K. (ed.) Proc. of the Ninth ACM SIGPLAN International Conference on Functional Programming (ICFP 2004), Snowbird, Utah, pp. 54–65 (September 2004)

Winskel, G.: Set theory for computer science. Unpublished lecture notes (2010), `http://www.cl.cam.ac.uk/~gw104/STfCS2010.pdf`

A Calculating the Abstract Transfer Function

Let P_i, ρ_e, and ρ_i be given.

$$
\begin{aligned}
& \alpha_\times \circ \alpha_\sim \circ (T(\gamma_\subset(P_i))) \circ \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i) && \text{(def. of } T\text{)} \\
= \; & \alpha_\times \circ \alpha_\sim\Big(\bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i))}} \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i) \;\sqcup\; [\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i))]\Big) \\
& && (\alpha_\times \circ \alpha_\sim \text{ a CJM}) \\
= \; & \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i))}} \alpha_\times \circ \alpha_\sim(\gamma_\sim \circ \gamma_\times(\rho_e, \rho_i) \;\sqcup\; [\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i))]) \\
& && (\alpha_\times \circ \alpha_\sim \text{ a CJM}) \\
= \; & \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i))}} \alpha_\times \circ \alpha_\sim \circ \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i) \;\sqcup\; \alpha_\times \circ \alpha_\sim([\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i))]) \\
& && \text{(Galois surjection)} \\
= \; & \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i))}} (\rho_e, \rho_i) \;\sqcup\; \alpha_\times \circ \alpha_\sim([\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e, \rho_i))]) && \text{(case analysis)}
\end{aligned}
$$

$$= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap XLam}} (\rho_e, \rho_i) \sqcup \alpha_\times \circ \alpha_\sim([\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))])$$

$$\sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap ILam}} (\rho_e, \rho_i) \sqcup \alpha_\times \circ \alpha_\sim([\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))]) \qquad \text{(def. of } \alpha_\sim\text{)}$$

$$= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap XLam}} (\rho_e, \rho_i) \sqcup \alpha_\times([\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))], \lambda x.\, \emptyset)$$

$$\sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap ILam}} (\rho_e, \rho_i) \sqcup \alpha_\times(\lambda x.\, \emptyset, [\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))]) \qquad \text{(def. of } \alpha_\times\text{)}$$

$$= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap XLam}} (\rho_e, \rho_i) \sqcup (\alpha_@ \circ \alpha_\cup([\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))]), \alpha_\cdot(\lambda x.\, \emptyset))$$

$$\sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap ILam}} (\rho_e, \rho_i) \sqcup (\alpha_@ \circ \alpha_\cup(\lambda x.\, \emptyset), \alpha_\cdot([\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))]))$$

(def. of $\alpha_\cup$)

$$= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap XLam}} (\rho_e, \rho_i) \sqcup (\alpha_@(\bigcup_{j \in [1;n]} E(t_j, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))), \alpha_\cdot(\lambda x.\, \emptyset))$$

$$\sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap ILam}} (\rho_e, \rho_i) \sqcup (\alpha_@(\emptyset), \alpha_\cdot([\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))])) \qquad (\alpha_@ \text{ a CJM})$$

$$= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap XLam}} (\rho_e, \rho_i) \sqcup (\bigcup_{j \in [1;n]} \alpha_@ \circ E(t_j, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)), \alpha_\cdot(\lambda x.\, \emptyset))$$

$$\sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap ILam}} (\rho_e, \rho_i) \sqcup (\alpha_@(\emptyset), \alpha_\cdot([\overline{x} \mapsto E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))])) \qquad \text{(def. of } \alpha_\cdot\text{)}$$

$$= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap XLam}} (\rho_e, \rho_i) \sqcup (\bigcup_{j \in [1;n]} \alpha_@ \circ E(t_j, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)), \lambda x.\, \alpha_@(\emptyset))$$

$$\sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap ILam}} (\rho_e, \rho_i) \sqcup (\alpha_@(\emptyset), [\overline{x} \mapsto \alpha_@ \circ E(\overline{t}, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i))]) \qquad \text{(Lemma 1)}$$

$$= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap XLam}} (\rho_e, \rho_i) \sqcup (\bigcup_{j \in [1;n]} \widehat{E}(t_j, \rho_e, \rho_i), \lambda x.\, \alpha_@(\emptyset))$$

$$\sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^\ell \in \gamma_\subset(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_\sim \circ \gamma_\times(\rho_e,\rho_i)) \cap ILam}} (\rho_e, \rho_i) \sqcup (\alpha_@(\emptyset), [\overline{x} \mapsto \widehat{E}(\overline{t}, \rho_e, \rho_i)]) \qquad \text{(def. of } \alpha_@\text{)}$$

$$
\begin{aligned}
&= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i)) \cap XLam}} (\rho_e, \rho_i) \sqcup \Big(\bigcup_{j \in [1;n]} \widehat{E}(t_j, \rho_e, \rho_i), \lambda x.\, \emptyset \Big) \\
&\quad \sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i)) \cap ILam}} (\rho_e, \rho_i) \sqcup (\emptyset, [\overline{x} \mapsto \widehat{E}(\overline{t}, \rho_e, \rho_i)]) && \text{(def. of } \sqcup\text{)} \\
&= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i)) \cap XLam}} (\rho_e \cup \bigcup_{j \in [1;n]} \widehat{E}(t_j, \rho_e, \rho_i), \rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i)) \cap ILam}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t}, \rho_e, \rho_i)]) && \text{(def. of } \alpha_{@}\text{)} \\
&= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i)) \cap XLam}} (\rho_e \cup \bigcup_{j \in [1;n]} \widehat{E}(t_j, \rho_e, \rho_i), \rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \alpha_{@} \circ E(t_0, \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i))}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t}, \rho_e, \rho_i)]) && \text{(Lemma 1)} \\
&= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in E(t_0, \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i)) \cap XLam}} (\rho_e \cup \bigcup_{j \in [1;n]} \widehat{E}(t_j, \rho_e, \rho_i), \rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0, \rho_e, \rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t}, \rho_e, \rho_i)]) && (\alpha_{@} \text{ monotone}) \\
&\sqsubseteq \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ \mathbf{xproc} \in \alpha_{@} \circ E(t_0, \gamma_{\sim} \circ \gamma_{\times}(\rho_e, \rho_i))}} (\rho_e \cup \bigcup_{j \in [1;n]} \widehat{E}(t_j, \rho_e, \rho_i), \rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0, \rho_e, \rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t}, \rho_e, \rho_i)]) && \text{(Lemma 1)} \\
&= \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ \mathbf{xproc} \in \widehat{E}(t_0, \rho_e, \rho_i)}} (\rho_e \cup \bigcup_{j \in [1;n]} \widehat{E}(t_j, \rho_e, \rho_i), \rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{(t_0\ t_1 \ldots t_n)^{\ell} \in \gamma_{\subset}(P_i) \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0, \rho_e, \rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t}, \rho_e, \rho_i)]) && \text{(Galois connection)} \\
&= \bigsqcup_{\substack{\alpha_{\subset}(\{(t_0\ t_1 \ldots t_n)^{\ell}\}) \subseteq P_i \\ \mathbf{xproc} \in \widehat{E}(t_0, \rho_e, \rho_i)}} (\rho_e \cup \bigcup_{j \in [1;n]} \widehat{E}(t_j, \rho_e, \rho_i), \rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{\alpha_{\subset}(\{(t_0\ t_1 \ldots t_n)^{\ell}\}) \subseteq P_i \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0, \rho_e, \rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t}, \rho_e, \rho_i)]) && \text{(def. of } \alpha_{\subset}\text{)}
\end{aligned}
$$

$$
\begin{aligned}
&= \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq XSexp \\ \mathbf{xproc} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e \cup \bigcup_{j\in[1;n]} \widehat{E}(t_j,\rho_e,\rho_i),\rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq P_i \\ \mathbf{xproc} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e \cup \bigcup_{j\in[1;n]} \widehat{E}(t_j,\rho_e,\rho_i),\rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq XSexp \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t},\rho_e,\rho_i)]) \\
&\quad \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq P_i \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t},\rho_e,\rho_i)]) \qquad \text{(Lemma 2)} \\
&\sqsubseteq \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq XSexp \\ \mathbf{xproc} \in (\rho_e \cup \{\mathbf{xproc}\} \cup Toplevel)}} (\rho_e \cup (\rho_e \cup \{\mathbf{xproc}\} \cup Toplevel),\rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq P_i \\ \mathbf{xproc} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e \cup \bigcup_{j\in[1;n]} \widehat{E}(t_j,\rho_e,\rho_i),\rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq XSexp \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in (\rho_e \cup \{\mathbf{xproc}\} \cup Toplevel)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto (\rho_e \cup \{\mathbf{xproc}\} \cup Toplevel)]) \\
&\quad \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq P_i \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t},\rho_e,\rho_i)]) \qquad \text{(simplify)} \\
&= (\rho_e \cup \{\mathbf{xproc}\} \cup Toplevel, \rho_i) \\
&\quad \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq P_i \\ \mathbf{xproc} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e \cup \bigcup_{j\in[1;n]} \widehat{E}(t_j,\rho_e,\rho_i),\rho_i) \\
&\quad \sqcup \bigsqcup_{(\lambda\, x_1 \ldots x_n.\ s)^{\ell} \in \rho_e \cup Toplevel} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto (\rho_e \cup \{\mathbf{xproc}\} \cup Toplevel)]) \\
&\quad \sqcup \bigsqcup_{\substack{\{(t_0\ t_1 \ldots t_n)^{\ell}\} \subseteq P_i \\ (\lambda\, x_1 \ldots x_n.\ s)^{\ell'} \in \widehat{E}(t_0,\rho_e,\rho_i)}} (\rho_e, \rho_i \ \dot\cup\ [\overline{x} \mapsto \widehat{E}(\overline{t},\rho_e,\rho_i)])
\end{aligned}
$$