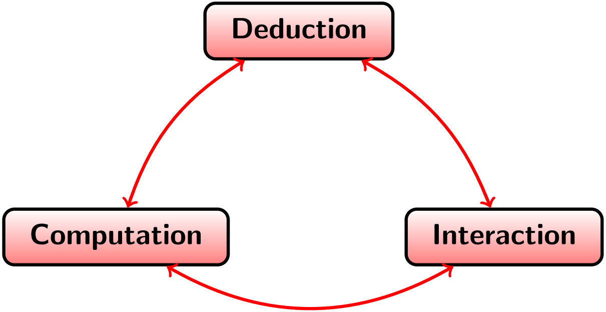
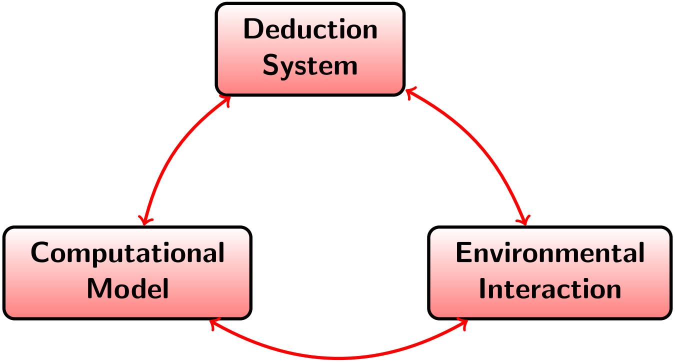
Modular operational nominal game semantics

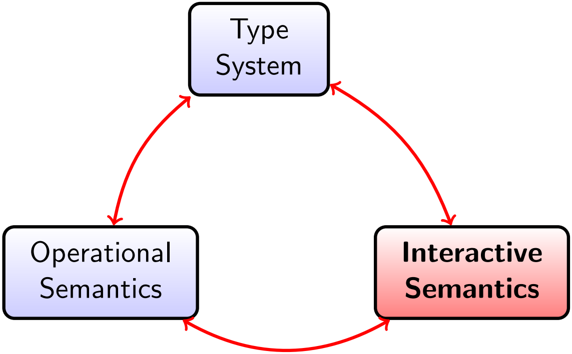
Guilhem Jaber (Univ. Nantes)

Scalp Meeting


Deduction
Computation
Interaction

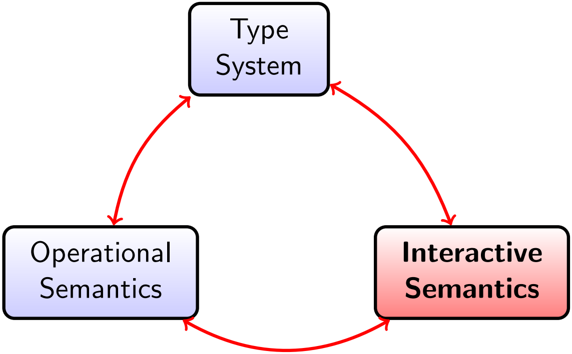

Deduction System
Computational Model
Environmental Interaction

In this talk



Type System
Operational Semantics
Interactive Semantics

In this talk

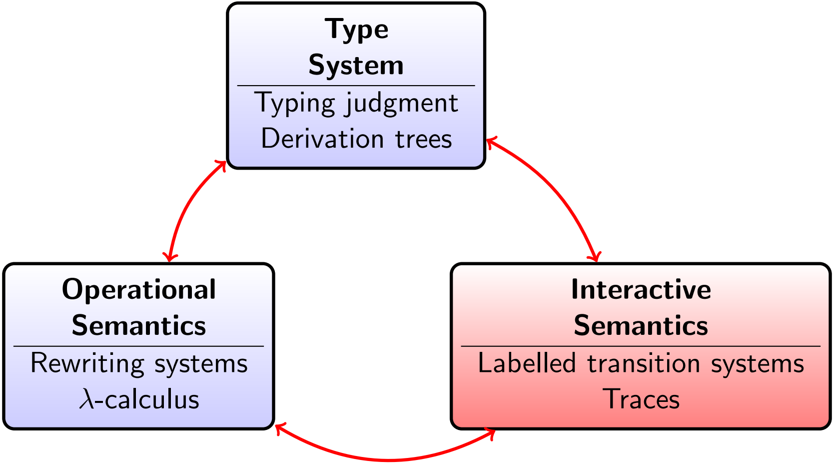


How to define interactive models of programming languages that are:

- abstract with respect to computation
- weakly coupled with the type system

Modularity

Structuring



What is a *good* interactive model ?

- A model that captures the observational power of the programming language.

- In a closed world:
  - $\rightsquigarrow$ captured via contextual equivalence;
  - $\rightsquigarrow$ Full abstraction.

How to define the observational power of a programming language ?

Via **polarization**:

- *interact* with negative ($\ominus$) values;
- *observe* positive ($\oplus$) values.

Operational Languages

$\mathrm{L}_{\mathsf{op}} = (\mathrm{Terms}, \mapsto_{\mathsf{op}})$ with *operational reduction relation*:

$$M \mapsto_{\mathsf{op}} N$$

Example for $\mathrm{L}_{\mathsf{cbv}}$ the pure λ-calculus in call-by-value

$$\begin{array}{llcl} \text{Values} & V, W & \triangleq & x \mid n \mid \lambda x.N \\ \text{Terms} & M, N & \triangleq & V \mid MN \\ \text{ECxts} & E, E' & \triangleq & \bullet \mid EM \mid VE \end{array}$$

$$E[(\lambda x.M)V] \mapsto_{\mathsf{op}} E[M\{x := V\}]$$

Names n are atoms used to represent the interaction with the outside world.

Lemma

A closed term of $\mathrm{L}_{\mathsf{cbv}}$ in normal form is either a value V or a head normal form $E[nV]$.

Contextual Equivalence

Definition

Two closed terms M, N are said to be *contextually equivalent* written $\Gamma \vdash M \simeq_{ctx} N : \theta$, when for all closed typed evaluation context $\varnothing \vdash E : \theta \rightsquigarrow \texttt{Unit}$ and substitution $\varnothing \vdash \gamma : \Gamma \rightharpoonup \mathrm{Values}$, we have $E[M\{\gamma\}]\Downarrow$ if and only if $E[N\{\gamma\}]\Downarrow$.

- Termination as observation
- restricted to non-binding contexts as with ciu-equivalence (Mason & Talcott);
- Contextual equivalence is the greatest adequate congruence (for evaluation contexts here).

Positive Types

We consider $\mathtt{L}^{ty}_{\mathsf{cbv}}$ that extends $\mathtt{L}_{\mathsf{cbv}}$ with:

- unit $()$ for the type `Unit`;
- pairs $\langle M, N \rangle$ for positive products type $\theta \times \theta'$;
- injections $\mathtt{inj}_1(M), \mathtt{inj}_2(M)$ for positive sum type $\theta + \theta'$.

Positive Types

We consider $\mathtt{L}^{ty}_{\mathsf{cbv}}$ that extends $\mathtt{L}_{\mathsf{cbv}}$ with:

- unit () for the type `Unit`;
- pairs $\langle M, N \rangle$ for positive products type $\theta \times \theta'$;
- injections $\mathtt{inj}_1(M), \mathtt{inj}_2(M)$ for positive sum type $\theta + \theta'$.

Positive values $\qquad A, B \triangleq n \mid \langle A, B \rangle \mid \mathtt{inj}_1(A) \mid \mathtt{inj}_2(A)$

Positive Types

We consider $\mathtt{L}^{ty}_{\mathsf{cbv}}$ that extends $\mathtt{L}_{\mathsf{cbv}}$ with:

- unit () for the type $\mathtt{Unit}$;
- pairs $\langle M, N\rangle$ for positive products type $\theta \times \theta'$;
- injections $\mathtt{inj}_1(M), \mathtt{inj}_2(M)$ for positive sum type $\theta + \theta'$.

Positive values $\quad A, B \triangleq n \mid \langle A, B\rangle \mid \mathtt{inj}_1(A) \mid \mathtt{inj}_2(A)$

Polarization process $V \mapsto_\oplus (A, \gamma)$:

$$\frac{}{f \mapsto_\oplus (g, [f \mapsto g])} \qquad \frac{}{\lambda x.M \mapsto_\oplus (f, [f \mapsto \lambda x.M])} \qquad \frac{}{() \mapsto_\oplus ((), \varepsilon)}$$

$$\frac{V \mapsto_\oplus (A, \gamma) \quad W \mapsto_\oplus (B, \gamma')}{\langle V, W\rangle \mapsto_\oplus (\langle A, B\rangle, \gamma \cdot \gamma')} \qquad \frac{V \mapsto_\oplus (A, \gamma)}{\mathtt{inj}_i(V) \mapsto_\oplus (\mathtt{inj}_i(A), \gamma)}$$

Lemma

If $V \mapsto_\oplus (A, \gamma)$ then $A\{\gamma\} = V$.

Operational Nominal Game Semantics

- provides trace-based abstractions to represent the interaction with the environment;
- introduces general reasoning principles to prove connections with the operational semantics and the type system;
- uses *Labelled Transition Systems* (LTS) as the basic blocs;
- inherits a sequential structure and a composition from game semantics;
- provides nominal resource usage control.

Traces

- The two players exchange *moves*, which are in one of four forms:

Move kind	P-question	P-answer	O-question	O-answer
Move form	$\overline{f}(A)$	$\overline{\mathrm{ret}}(A)$	$f(A)$	$\mathrm{ret}(A)$

- *Traces* t are sequences of moves.
- *positive values* A are the *observable* terms.

The Operational Game Semantics Recipe

Start with a Program (Proponent)

1. Compute its normal form via $\mapsto_{\mathsf{op}}$;
2. Polarize the normal form via $\mapsto_{\oplus}$;
3. Check the correctness of the polarization via the type system ($\vdash_{\oplus}$);
4. Perform the action p from the polarized normal form;
5. Let the Environment (Opponent) performs back an action o;
6. Check the correctness of the polarization of o via $\vdash_{\oplus}$;
7. Trigger the computation associated by this action;
8. Repeat to point 1.

Introducing the OGS LTS

$$\mathcal{L}_{\mathsf{ogs}} = (\mathrm{Confs}_{\mathsf{ogs}}, \overset{\mathsf{a}}{\mapsto}_{\mathsf{ogs}}, \mathsf{a} \in \mathrm{Moves} \cup \{\mathsf{op}\})$$

is the product $\mathcal{L}_{\mathsf{I}} \ltimes \mathcal{L}_{\mathsf{Ty}}$ of:

- *Interactive LTS* $\mathcal{L}_{\mathsf{I}}$ (with $\mathbb{I} \in \mathrm{Confs}_{\mathsf{I}}$);
- *Type LTS* $\mathcal{L}_{\mathsf{Ty}}$ (with $\mathbb{S} \in \mathrm{Confs}_{\mathsf{Ty}}$).
- They share the same actions a;
- *Typing relation* $\mathbb{I} \triangleright \mathbb{S}$
- Configurations $\mathbb{G} \in \mathrm{Confs}_{\mathsf{ogs}}$ such that $\mathbb{G} = (\mathbb{I}; \mathbb{S})$ with $\mathbb{I} \triangleright \mathbb{S}$.

$$\frac{\mathbb{I} \triangleright \mathbb{S} \qquad \mathbb{I} \xrightarrow{\mathsf{a}}_{\mathsf{I}} \mathbb{J} \qquad \mathbb{S} \xrightarrow{\mathsf{a}}_{\mathsf{Ty}} \mathbb{T} \qquad \mathbb{J} \triangleright \mathbb{T}}{(\mathbb{I}; \mathbb{S}) \overset{\mathsf{a}}{\mapsto}_{\mathsf{ogs}} (\mathbb{J}; \mathbb{T})}$$

Interactive LTS for $\mathrm{L}_{\mathsf{cbv}}$

"Abstract machines for Interaction"

$$\mathcal{L}_{\mathsf{I}} = (\mathrm{Confs}_{\mathsf{I}}, \xrightarrow{\mathsf{a}}_{\mathsf{I}}, \mathsf{a} \in \mathrm{Moves} \cup \{\mathsf{op}\})$$

- $\mapsto_{\mathsf{op}}$ is embedded into $\xrightarrow{\mathsf{op}}_{\mathsf{I}}$
- visible actions are moves;
- configurations $\mathbb{I}$ are either active $\langle M; \sigma; \gamma \rangle$ or passive $\langle \sigma; \gamma \rangle$;
- γ is a list of substitutions from names to values;
- σ is a stack of evaluation contexts;

The Interactive LTS for $\mathbb{L}_{\mathsf{cbv}}$

$$
\begin{array}{l|lll}
\text{op} & \langle M; \sigma; \gamma \rangle & \xrightarrow{\mathsf{op}}_{\mathsf{I}} & \langle N; \sigma; \gamma \rangle \text{ when } M \mapsto_{\mathsf{op}} N \\
\text{PQ} & \langle E[fV]; \sigma; \gamma \rangle & \xrightarrow{\overline{f}(g)}_{\mathsf{I}} & \langle E :: \sigma; \gamma \cdot [g \mapsto V] \rangle \\
\text{PA} & \langle V; \sigma; \gamma \rangle & \xrightarrow{\overline{\mathsf{ret}}(f)}_{\mathsf{I}} & \langle \sigma; \gamma \cdot [f \mapsto V]] \rangle \\
\text{OQ} & \langle \sigma; \gamma \rangle & \xrightarrow{f(g)}_{\mathsf{I}} & \langle \gamma(f) g; \sigma; \gamma \rangle \\
\text{OA} & \langle E :: \sigma; \gamma \rangle & \xrightarrow{\mathsf{ret}(f)}_{\mathsf{I}} & \langle E[f]; \sigma; \gamma \rangle
\end{array}
$$

The Interactive LTS for $\mathtt{L}^{ty}_{\mathsf{cbv}}$

$$
\begin{array}{l|lcl}
\mathsf{op} & \langle M; \sigma; \gamma \rangle & \xrightarrow{\mathsf{op}}_{\mathsf{I}} & \langle N; \sigma; \gamma \rangle \text{ when } M \mapsto_{\mathsf{op}} N \\
\mathsf{PQ} & \langle E[fV]; \sigma; \gamma \rangle & \xrightarrow{\overline{f}(A)}_{\mathsf{I}} & \langle E :: \sigma; \gamma \cdot \gamma' \rangle \text{ when } V \mapsto_{\oplus} (A, \gamma') \\
\mathsf{PA} & \langle V; \sigma; \gamma \rangle & \xrightarrow{\overline{\mathsf{ret}}(A)}_{\mathsf{I}} & \langle \sigma; \gamma \cdot \gamma'] \rangle \text{ when } V \mapsto_{\oplus} (A, \gamma') \\
\mathsf{OQ} & \langle \sigma; \gamma \rangle & \xrightarrow{f(A)}_{\mathsf{I}} & \langle \gamma(f) A; \sigma; \gamma \rangle \\
\mathsf{OA} & \langle E :: \sigma; \gamma \rangle & \xrightarrow{\mathsf{ret}(A)}_{\mathsf{I}} & \langle E[A]; \sigma; \gamma \rangle
\end{array}
$$

Typing LTS

$$\mathcal{L}_{\mathsf{Ty}} = (\mathrm{Confs}_{\mathsf{Ty}}, \xrightarrow{\mathsf{a}}_{\mathsf{Ty}}, \mathsf{a} \in \mathrm{Moves} \cup \{\mathsf{op}\})$$

- configurations $\mathbb{S}$ keep tracks of typing of names;
- transition checks typing constraints on positive values exchanged using $\vdash_{\oplus}$;
- mainly to control Opponent behavior;
- but also to guide Proponent polarization $\mapsto_{\oplus}$ in presence of polymorphism.

The Typing LTS for $\mathrm{L}^{ty}_{\mathrm{cbv}}$

$$
\begin{array}{l|l}
\text{op} & \langle \Delta_O \vdash \theta; \Sigma; \Delta_P \rangle \quad \xrightarrow{\text{op}}_{\mathrm{I}} \quad \langle \Delta_O \vdash \theta; \Sigma; \Delta_P \rangle \\
\text{PQ} & \langle \Delta_O \vdash \theta; \Sigma; \Delta_P \rangle \quad \xrightarrow{\overline{f}(A)}_{\mathrm{I}} \quad \langle \Delta_O \vdash \Sigma'; \Delta_P \cdot \Delta'_P \rangle \\
 & \text{ when } \Delta'_P \vdash_{\oplus} A : \theta_1 \text{ with } \Delta_O(f) = \theta_1 \to \theta_2 \text{ and } \Sigma' = (\theta_2 \rightsquigarrow \theta) :: \Sigma \\
\text{PA} & \langle \Delta_O \vdash \theta; \Sigma; \Delta_P \rangle \quad \xrightarrow{\overline{\text{ret}}(A)}_{\mathrm{I}} \quad \langle \Delta_O \vdash \Sigma; \Delta_P \cdot \Delta'_P \rangle, \\
 & \text{ when } \Delta'_P \vdash_{\oplus} A : \theta \\
\text{OQ} & \langle \Delta_O \vdash \Sigma; \Delta_P \rangle \quad \xrightarrow{f(A)}_{\mathrm{I}} \quad \langle \Delta_O \cdot \Delta'_O \vdash \theta'; \Sigma; \Delta_P \rangle \\
 & \text{ when } \Delta'_O \vdash_{\oplus} A : \theta \text{ with } \Delta_P(f) = \theta \to \theta' \\
\text{OA} & \langle \Delta_O \vdash \Sigma; \Delta_P \rangle \quad \xrightarrow{\text{ret}(A)}_{\mathrm{I}} \quad \langle \Delta_O \cdot \Delta'_O \vdash \theta'; \Sigma'; \Delta_P \rangle \\
 & \text{ when } \Delta'_O \vdash_{\oplus} A : \theta \text{ and } \Sigma' = (\theta \rightsquigarrow \theta') :: \Sigma
\end{array}
$$

Scaling to rich languages

- Higher-order references;
- Control operators;
- Parametric polymorphism;
- Private/disclosed nominal resources;
- Dynamic sealing.

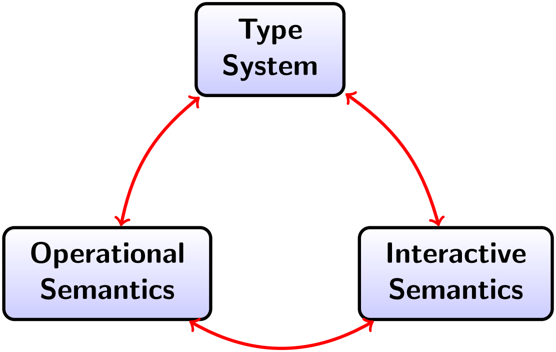
Teasing

Fully-abstract compilation from System F to the untyped cryptographic λ-calculus

(j.w.w. N. Tzevelekos)

- Correcting the compiler proposed by Sumii & Pierce (2000)
- via a *freshness check*
- designed from a double polarization interpretation of System F
- avoiding the counter-example provided by Devriese, Patrignani & Piessens (2018)

I need help finding my way through polarization...



Parametric polymorphism

Logical Relations:

- On denotational models (Reynolds; Plotkin);
- Logical systems (Plotkin-Abadi; Abadi, Curien & Cardelli)
- Dependent types (Berardi; Keller & Lasson)
- Operational logical relations (Pitts)
- Type system (Harper & Sterling).

Operational equivalence

- Strachey-Equivalence: type-erased terms are $\beta\eta$-equivalent
- **Contextual equivalence**

Interactive equivalence

- **Fully-abstract compilation** (Sumii & Pierce)
- Bisimulation (Lassen & Levy)
- **Game models** (Laird; J. & Tzevelekos)

Second-order typing

$$\frac{\Gamma, X : \mathtt{Type} \vdash M : \theta}{\Gamma \vdash \Lambda X.M : \forall X.\theta} \qquad \frac{\Gamma \vdash M : \forall X.\theta}{\Gamma \vdash M\theta : \theta'\{X := \theta\}}$$

$$\frac{\Gamma \vdash M : \theta\{X := \theta'\}}{\Gamma \vdash \langle \theta'; M \rangle : \exists X.\theta}$$

$$\frac{\Gamma \vdash M : \exists X.\theta \qquad \Gamma, X : \mathtt{Type}, x : \theta \vdash N : \theta'}{\Gamma \vdash \mathtt{match}\ M\ \mathtt{with}\ (X, x) \Rightarrow N : \theta'}$$

Polarization of type variables

- $X^{\oplus}$ when Proponent choose the type associated to X;
- $X^{\ominus}$ when Opponent choose the type associated to X.

Formally, a type translation $\mathsf{pol}_\eta(\cdot)^\kappa : \mathrm{Types} \to \mathrm{Types}$, for $\kappa \in \{\oplus, \ominus\}$, defined as

$$\begin{array}{rcl}
\mathsf{pol}_\eta(\theta \to \theta')^\kappa & \triangleq & \mathsf{pol}_\eta(\theta)^{\kappa^\perp} \to \mathsf{pol}_\eta(\theta')^\kappa \\
\mathsf{pol}_\eta(\theta \times \theta')^\kappa & \triangleq & \mathsf{pol}_\eta(\theta)^\kappa \times \mathsf{pol}_\eta(\theta')^\kappa \\
\mathsf{pol}_\eta(\theta + \theta')^\kappa & \triangleq & \mathsf{pol}_\eta(\theta)^\kappa + \mathsf{pol}_\eta(\theta')^\kappa \\
\mathsf{pol}_\eta(\forall X.\theta)^\kappa & \triangleq & \forall X.\mathsf{pol}_{\eta\cdot[X \mapsto \kappa^\perp]}(\theta)^\kappa \\
\mathsf{pol}_\eta(\exists X.\theta)^\kappa & \triangleq & \exists X.\mathsf{pol}_{\eta\cdot[X \mapsto \kappa]}(\theta)^\kappa \\
\mathsf{pol}_\eta(X)^\kappa & \triangleq & \eta(X)
\end{array}$$

Positive Typing for Polymorphism

- Typing judgment $\vdash_{\oplus}^{\mathsf{X}}$ indexed by player $\mathsf{X} \in \{\mathsf{P}, \mathsf{O}\}$.
- Splitted typing contexts $\Delta_{\mathsf{O}}|\Delta_{\mathsf{P}}$.

$$\frac{\Delta_{\mathsf{O}}(p) = \alpha}{\Delta_{\mathsf{O}}|\Delta_{\mathsf{P}} \vdash_{\oplus}^{\mathsf{P}} p : \alpha^{\ominus}} \qquad \frac{\Delta_{\mathsf{P}}(p) = \alpha}{\Delta_{\mathsf{O}}|\Delta_{\mathsf{P}} \vdash_{\oplus}^{\mathsf{O}} p : \alpha^{\oplus}}$$

$$\frac{p \in \mathrm{PNames}\backslash\mathrm{dom}(\Delta_{\mathsf{O}} \cdot \Delta_{\mathsf{P}})}{\Delta_{\mathsf{O}}|\Delta_{\mathsf{P}} \vdash_{\oplus}^{\mathsf{P}} \nu p : \alpha^{\oplus}} \qquad \frac{p \in \mathrm{PNames}\backslash\mathrm{dom}(\Delta_{\mathsf{O}} \cdot \Delta_{\mathsf{P}})}{\Delta_{\mathsf{O}}|\Delta_{\mathsf{P}} \vdash_{\oplus}^{\mathsf{O}} \nu p : \alpha^{\ominus}}$$

$$\frac{\Delta_{\mathsf{O}}|\Delta_{\mathsf{P}}, \alpha : \mathtt{Type} \vdash_{\oplus}^{\mathsf{P}} A : \theta'\{X := \alpha\}}{\Delta_{\mathsf{O}}|\Delta_{\mathsf{P}} \vdash_{\oplus}^{\mathsf{P}} \langle \nu\alpha; A \rangle : \exists X.\theta'}$$

$$\frac{\Delta_{\mathsf{O}}, \alpha : \mathtt{Type}|\Delta_{\mathsf{P}} \vdash_{\oplus}^{\mathsf{O}} A : \theta'\{X := \alpha\}}{\Delta_{\mathsf{O}}|\Delta_{\mathsf{P}} \vdash_{\oplus}^{\mathsf{O}} \langle \nu\alpha; A \rangle : \exists X.\theta'}$$

Memoryful programs

$$\mathrm{L}_{\mathsf{op}} = (\mathrm{Terms}, \mathrm{Stores}, \mapsto_{\mathsf{op}}) \text{ with } \textit{operational reduction relation}:$$

$$(M, \xi) \mapsto_{\mathsf{op}} (N, \zeta)$$

Example for the ν-calculus, with dynamic creation of *atoms an* stored in the set S:

$$\begin{array}{lcl}
(E[(\lambda x.M)V], S) & \mapsto_{\mathsf{op}} & (E[M\{x := V\}], S) \\
(E[\mathtt{if\ true\ then}\ N_1\ \mathtt{else}\ N_2], S) & \mapsto_{\mathsf{op}} & (N_1, , S) \\
(E[\mathtt{if\ false\ then}\ N_1\ \mathtt{else}\ N_2], S) & \mapsto_{\mathsf{op}} & (N_2, S) \\
(E[\mathtt{a} = \mathtt{a}], S) & \mapsto_{\mathsf{op}} & (E[\mathtt{true}], S) \\
(E[\mathtt{a} = \mathtt{a}'], S) & \mapsto_{\mathsf{op}} & (E[\mathtt{false}], S) \\
(E[\mathtt{new}\ x\ \mathtt{in}\ M], S) & \mapsto_{\mathsf{op}} & (E[M\{x := \mathtt{a}\}], S \uplus \{\mathtt{a}\})
\end{array}$$

A taxonomy of resource usage

We consider the following properties on named resources:

- calleable (with/without well-bracketed discipline);
- disclosable / scoped (visible) / storeable;
- affine / duplicable / persistent;
- readable / sealed / writeable;
- internally bindable;
- internally allocatable/disallocable
- distinguishable;
- typed.

History LTS

$$\mathcal{L}_{\mathsf{H}} = (\mathrm{Confs}_{\mathsf{H}}, \overset{\mathsf{a}}{\mapsto}_{\mathsf{H}}, \mathsf{a} \in \mathrm{Moves} \cup \{\mathsf{op}\})$$

- configurations keep track of the usage of names: disclosed(set),well-bracketed(stack) or scoped(tree) discipline;
- transitions check that they are respected by Opponent.

Ressource LTS for atom generation

- S is the set of atoms known by P;
- D is the set of atoms known by both P and O.

$$
\begin{array}{l|l}
\text{op} & \langle M; S; \sigma; \gamma \rangle \quad \triangleright \langle D \rangle \xrightarrow{\text{op}}_{\mathsf{I}} \quad \langle N; S'; \sigma; \gamma \rangle \quad \triangleright \quad \langle D \rangle \\
& \text{when } (M, S) \mapsto_{\text{op}} (N, S') \\
\text{PQ} & \langle E[xV]; S; \sigma; \gamma \rangle \triangleright \langle D \rangle \xrightarrow{\overline{f}(A)}_{\mathsf{I}} \quad \langle S; E :: \sigma; \gamma \cdot \gamma' \rangle \ \triangleright \langle D \cup D' \rangle \\
& \text{when } V \mapsto_{\oplus} (A, \gamma') \\
\text{PA} & \langle V; S; \sigma; \gamma \rangle \quad \triangleright \langle D \rangle \xrightarrow{\overline{\text{ret}}(A)}_{\mathsf{I}} \quad \langle S; \sigma; \gamma \cdot \gamma' \rangle \quad \triangleright \langle D \cup D' \rangle, \\
& \text{when } V \mapsto_{\oplus} (A, \gamma') \\
\text{OQ} & \langle S; \sigma; \gamma \rangle \quad \triangleright \langle D \rangle \xrightarrow{f(A)}_{\mathsf{I}} \quad \langle VA; S \cup D'; \sigma; \gamma \rangle \ \triangleright \langle D \cup D' \rangle \\
& \text{when } \gamma(f) = V \\
\text{OA} & \langle S; E :: \sigma; \gamma \rangle \quad \triangleright \langle D \rangle \xrightarrow{\text{ret}(A)}_{\mathsf{I}} \langle E[A]; S \cup D'; \sigma; \gamma \rangle \triangleright \langle D \cup D' \rangle
\end{array}
$$

With D' the set of atoms of A not in D. In the Opponent transitions, we suppose the following *Non-omniscient* condition:

$$D' \cap S = \varnothing$$

Composition/Interaction operational reduction

- a *CIO reduction relation* $\mapsto_{\mathsf{cio}}$ extend $\rightarrow_{\mathsf{l}}$ into parallel composition plus hiding.
- configurations $\mathbb{D}, \mathbb{D}' \in \mathrm{Confs}_{\mathsf{cio}} \triangleq \{\nu\big(\mathbb{G}||\mathbb{H}\big) \mid \mathbb{G}\perp\mathbb{H}\}$
- $\mapsto_{\mathsf{cio}}$ defined by the following rules:

$$\frac{\mathbb{G} \xrightarrow{\mathsf{m}}_{\mathsf{l}} \mathbb{G}' \quad \mathbb{H} \xrightarrow{\mathsf{m}^{\perp}}_{\mathsf{l}} \mathbb{H}'}{\nu\big(\mathbb{G}||\mathbb{H}\big) \mapsto_{\mathsf{cio}} \nu\big(\mathbb{G}'||\mathbb{H}'\big)}$$

Adequacy via abstract machines

Introduce an operational reduction reduction

$$(\mathrm{Confs}_{\mathsf{bop}}, \mapsto_{\mathsf{bop}})$$

together with two functional bisimulation
$\kappa_{\mathsf{bop}} : \mathrm{Confs}_{\mathsf{bop}} \to \mathrm{Terms} \times \mathrm{Stores}$ and $\kappa_{\mathsf{cio}} : \mathrm{Confs}_{\mathsf{cio}} \to \mathrm{Confs}_{\mathsf{bop}}$.
In $\mathrm{L}_{\mathsf{cbv}}$:

- configurations are of the shape $(M; \sigma; \delta)$
- $\mapsto_{\mathsf{bop}}$ defined by the following rules:

$$\frac{V \mapsto_{\oplus} (A, \delta')}{(E[fV]; \sigma; \delta) \mapsto_{\mathsf{bop}} (\delta(f)A; E :: \sigma; \delta \cdot \delta')}$$

$$\frac{V \mapsto_{\oplus} (A, \delta')}{(V; E :: \sigma; \delta) \mapsto_{\mathsf{bop}} (E[A]; \sigma; \delta \cdot \delta')} \qquad \frac{M \mapsto_{\mathsf{op}} N}{(M; \sigma; \delta) \mapsto_{\mathsf{bop}} (N; \sigma; \delta)}$$

Soundness of trace equivalence

Suppose we observe only booleans.

Definition

We write $(\mathbb{G}, \mathbb{H}) \in \perp\!\!\!\perp$ when there exists a complete traces t such that $\mathsf{t} \in \mathsf{CTr}(\mathbb{G})$ and $\mathsf{t}^{\perp} \cdot \overline{\mathsf{ret}}(\mathtt{b}) \in \mathsf{CTr}(\mathbb{H})$ with $\mathtt{b} \in \{\mathtt{true}, \mathtt{false}\}$.

Lemma

$(\mathbb{G}, \mathbb{H}) \in \perp\!\!\!\perp$ *if and only if* $\nu\big(\mathbb{G} || \mathbb{H}\big) \Downarrow$.

Lemma

If $(\mathbb{G}, \mathbb{H}) \in \perp\!\!\!\perp$ *and* $\mathbb{G} \simeq_{tr} \mathbb{G}'$ *then* $(\mathbb{G}', \mathbb{H}) \in \perp\!\!\!\perp$.

Theorem

Taking two terms M, N *such that* $\Gamma \vdash M, N : \theta$, *if* $\iota(\Gamma \vdash M : \theta) \simeq_{tr} \iota(\Gamma \vdash N : \theta)$ *then* $\Gamma \vdash M \simeq_{ctx} N : \theta$.

Full-abstraction of trace equivalence

Theorem

Taking two terms M, N such that $\Gamma \vdash M, N : \theta$, if $\Gamma \vdash M \simeq_{ctx} N : \theta$ then $\iota(\Gamma \vdash M : \theta) \simeq_{ctr} \iota(\Gamma \vdash N : \theta)$.

- Need a definability result to transform a trace t into a term that generates this trace;
- Holds only in presence of some memory (one integer mutable memory cell).
- Need to relax the notion of trace equivalence to *complete trace equivalence* $\simeq_{ctr}$ when contexts are control operators free.

Fully-abstract compilation

Definition

$(\!|\cdot|\!) : \mathrm{L}_1 \to \mathrm{L}_2$ is fully abstract if for all terms M, N of L_1, we have $M \simeq_{ctx}^{\mathrm{L}_1} N$ iff $(\!|M|\!) \simeq_{ctx}^{\mathrm{L}_2} (\!|N|\!)$.

Suppose L_2-contexts are more powerful than L_1-contexts:

- $\mapsto_{\oplus}^{2}$ is contained in $\mapsto_{\oplus}^{1}$: L_2-contexts can observe more.
- $\vdash_{\oplus}^{1}$ is contained in $\vdash_{\oplus}^{2}$: L_2-contexts can interact more.

The compilation $(\!|\cdot|\!)$ embeds some runtime checks at the interaction points (normal forms) to:

- perform on Proponent interactions the extra opacification steps needed that $\mapsto_{\oplus}^{1}$ does but not $\mapsto_{\oplus}^{2}$;
- reject on Opponent interactions the positive values validated by $\vdash_{\oplus}^{2}$ but not by $\vdash_{\oplus}^{1}$.

Proving full-abstraction results for compilers

Theorem

$(\!|\cdot|\!) : \mathtt{L}_1 \to \mathtt{L}_2$ *is fully abstract when:*

- $\simeq_{tr}^{\mathtt{L}_1}$ *is fully abstract wrt* $\simeq_{ctx}^{\mathtt{L}_1}$
- $\simeq_{tr}^{\mathtt{L}_2}$ *is sound wrt* $\simeq_{ctx}^{\mathtt{L}_2}$
- $(\!|\cdot|\!)$ *induces a bisimulation between* $\mathcal{L}_{\mathsf{OGS}}(\mathtt{L}_1)$ *and* $\mathcal{L}_{\mathsf{OGS}}(\mathtt{L}_2)$.

Presentation of the context

In 2000, Sumii & Pierce proposed a compilation scheme $(\!|\cdot|\!) : \mathsf{F} \to \mathtt{L_c}$ between:

- the (second-order) polymorphic λ-calculus F;
- the cryptographic λ-calculus $\mathtt{L_c}$, a simply-typed λ-calculus equipped with some dynamic sealing properties

The compilation scheme inserts some runtime to enforce dynamically the parametricity properties provided by the polymorphic type system.

Sumii & Pierce conjectured $(\!|\cdot|\!)$ to be fully-abstract.

Cryptographic λ-calculus

- Sealed values $\{V\}_\sigma$ with a seal σ
- Dynamic seal creation $\texttt{newseal}\ \sigma\ \texttt{in}\ M$
- Unsealing:

$$\texttt{match}\ \{V\}_\sigma\ \texttt{with}\ \begin{array}{l} |(\sigma', x) \ \Rightarrow M \\ |\mathit{wrong} \Rightarrow N \end{array} \ \mapsto_{\mathsf{op}} \begin{cases} M\{x := V\} \text{ when } \sigma = \sigma' \\ N \text{ otherwise} \end{cases}$$

- Type $\texttt{Seal}_\theta$ for seals that can be used only on values of type θ
  - $\leadsto$ needed to ensure type soundness.

Embedding F into $\mathtt{L_c}$

Type Erasure $\wr \cdot \wr : \mathsf{F} \to \mathtt{L_c}$

- on terms: remove type annotations (from Church to Curry-style);
- on types: remove second-order types:

$$
\begin{array}{lcl}
\wr \forall X.\theta \wr & \triangleq & \mathtt{Unit} \to \wr\, \theta \wr \\
\wr \exists X.\theta \wr & \triangleq & \mathtt{Unit} \times \wr\, \theta \wr \\
\wr \theta \to \theta' \wr & \triangleq & \wr \theta \,\wr \to \wr\, \theta' \wr \\
\wr X^{\oplus} \wr & \triangleq & \mathtt{Bytes} \\
\wr X^{\ominus} \wr & \triangleq & \mathtt{Bytes}
\end{array}
$$

Sumii & Pierce compilation scheme (I/III)

$(\!|\cdot|\!) : \mathsf{F} \to \mathtt{L_c}$ embeds some runtime checks at the interaction points:

- for Player transition:
  - at type $X^{\oplus}$: it seals using σ_X the value exchanged
  - at type $X^{\ominus}$: it does nothing.
- for Opponent transition:
  - at type $X^{\oplus}$: it unseals using σ_X the value provided
  - at type $X^{\ominus}$: it does nothing.

Sumii & Pierce compilation scheme (II/III)

$$
\begin{aligned}
\mathrm{protect}^{Seal}_{\eta,\mathrm{Unit}}\, x &\triangleq x \\
\mathrm{protect}^{Seal}_{\eta,\theta_1\times\theta_2}\, x &\triangleq \mathrm{let}\, x_1 = \pi_1(x)\, \mathrm{in}\, \mathrm{let}\, x_2 = \pi_2(x)\, \mathrm{in}\, \langle\, \mathrm{protect}^{Seal}_{\eta,\theta_1}\, x_1, \mathrm{protect}^{Seal}_{\eta,\theta_2} \\
\mathrm{protect}^{Seal}_{\eta,\theta\to\theta'}\, x &\triangleq \lambda y.\, \mathrm{let}\, z = x(\mathrm{confine}^{Seal}_{\eta,\theta}\, y)\, \mathrm{in}\, \mathrm{protect}^{Seal}_{\eta,\theta'}\, z \\
\mathrm{protect}^{Seal}_{\eta,\forall X.\theta}\, x &\triangleq \lambda\_.\, \mathrm{let}\, y = x()\, \mathrm{in}\, \mathrm{protect}^{Seal}_{\eta,\theta\{X:=\alpha^-\}} \\
\mathrm{protect}^{Seal}_{\eta,\exists X.\theta}\, x &\triangleq \mathrm{let}\, y = \pi_2(x)\, \mathrm{in}\, \nu\alpha.\langle(), \mathrm{protect}^{Seal}_{\eta\uplus\{\alpha\},\theta\{X:=\alpha^+\}}\, y\rangle \\
\mathrm{protect}^{Seal}_{\eta,\alpha^\oplus}\, x &\triangleq \mathrm{seal}_\alpha x \\
\mathrm{protect}^{Seal}_{\eta,\alpha^\ominus}\, x &\triangleq x
\end{aligned}
$$

Sumii & Pierce compilation scheme (III/III)

$$
\begin{aligned}
\mathsf{confine}^{Seal}_{\eta,\mathrm{Unit}}\, x &\triangleq x \\
\mathsf{confine}^{Seal}_{\eta,\theta_1\times\theta_2}\, x &\triangleq \mathsf{let}\, x_1 = \pi_1(x)\, \mathsf{in}\, \mathsf{let}\, x_2 = \pi_2(x)\, \mathsf{in}\, \langle\, \mathsf{confine}^{Seal}_{\eta,\theta_1}\, x_1, \mathsf{confine}^{Seal}_{\eta,\theta_2} \\
\mathsf{confine}^{Seal}_{\eta,\theta\to\theta'}\, x &\triangleq \lambda y.\, \mathsf{let}\, z = x(\mathsf{protect}^{Seal}_{\eta,\theta}\, y)\, \mathsf{in}\, \mathsf{confine}^{Seal}_{\eta,\theta'}\, z \\
\mathsf{confine}^{Seal}_{\eta,\forall X.\theta}\, x &\triangleq \lambda\_.\, \mathsf{let}\, y = x()\, \mathsf{in}\, \nu\alpha.\, \mathsf{confine}^{Seal}_{\eta\uplus\{\alpha\},\theta\{X:=\alpha^{\oplus}\}}\, y \\
\mathsf{confine}^{Seal}_{\eta,\exists X.\theta}\, x &\triangleq \mathsf{let}\, y = \pi_2(x)\, \mathsf{in}\, \langle(), \mathsf{confine}^{Seal}_{\eta,\theta\{X:=\alpha^{-}\}}\, y\rangle \\
\mathsf{confine}^{Seal}_{\eta,\alpha^{\oplus}}\, x &\triangleq \mathsf{unseal}_{\alpha} x \\
\mathsf{confine}^{Seal}_{\eta,\alpha^{\ominus}}\, x &\triangleq x
\end{aligned}
$$

Fully-abstract compilation ?

In 2018, Devriese, Patrignani & Piessens provide a counterexample to the fully-abstract conjecture based on the universal type:

$$\exists Y.\forall X.(X^{\ominus} \to Y^{\oplus}) \times (Y^{\oplus} \to X^{\ominus})$$

The pair $\langle (), \lambda\_.\langle \lambda x.x, \lambda x.x \rangle \rangle$ can fake the runtime check to pretend to be of this type.

Sumii & Pierce compilation scheme

$(\!|\cdot|\!) : \mathsf{F} \rightarrow \mathtt{L_c}$ embeds some runtime checks at the interaction points:

- for Player transition:
  - at type $X^{\oplus}$: it seales using σ_X the value exchanged
  - at type $X^{\ominus}$: it does nothing.

- for Opponent transition:
  - at type $X^{\oplus}$: it unseales using σ_X the value provided
  - **at type $X^{\ominus}$: it does nothing.**

Fully-abstract compilation

Theorem

$(\!|\cdot|\!) : \mathsf{F} \to \mathsf{L_c}$ *is fully abstract if for all terms* M, N *of* F, *we have* $M \simeq^{\mathsf{F}}_{tr} N$ *iff* $(\!|M|\!) \simeq^{\mathsf{L_c}}_{tr} (\!|N|\!)$.

$\mathsf{L_c}$-contexts are more powerful than F-contexts:

- $\mapsto^{\mathsf{L_c}}_{\oplus}$ is embedded in $\mapsto^{\mathsf{F}}_{\oplus}$: F-contexts can observe more.
- $\vdash^{\mathsf{F}}_{\oplus}$ is embedded in $\vdash^{\mathsf{L_c}}_{\oplus}$: $\mathsf{L_c}$-contexts can interact more.

The compilation $(\!|\cdot|\!)$ embeds some runtime checks at the interaction points (normal forms) to:

- perform on Proponent interactions the extra abstraction steps needed that $\mapsto^{\mathsf{F}}_{\oplus}$ does but not $\mapsto^{\mathsf{L_c}}_{\oplus}$
- **reject on Opponent interactions the abstract values validated by $\vdash^{\mathsf{L_c}}_{\oplus}$ but not by $\vdash^{\mathsf{F}}_{\oplus}$.**

Sumii & Pierce compilation scheme *fixed*

$(\!|\cdot|\!) : \mathsf{F} \to \mathtt{L_c}$ embeds some runtime checks at the interaction points:

- for Player transition:
  - at type $X^{\oplus}$: it seales using σ_X the value exchanged
  - at type $X^{\ominus}$: it does nothing.
- for Opponent transition:
  - at type $X^{\oplus}$: it unseales using σ_X the value provided
  - *at type $X^{\ominus}$: it checks for freshness of the value provided.*

Polarizing the cryptographic λ-calculus

- $\lambda^{\sigma\oplus/\ominus}$: Add polarization annotations $\texttt{Bytes}^{\oplus}, \texttt{Bytes}^{\ominus}$ to $\mathtt{L_c}$
  - $\rightsquigarrow$ positive when we can unseal the value;
  - $\rightsquigarrow$ negative when we cannnot.
- Polarization may evolve from $\oplus$ to $\ominus$ during the interaction !
- Use polarity information to define $\vdash_{\oplus}^{\lambda^{\sigma\oplus/\ominus}}$
- Type Erasure $\wr \cdot \int : \mathsf{F} \to \lambda^{\sigma\oplus/\ominus}$:

$$\begin{array}{lcl} \wr X^{\oplus} \int & \triangleq & \texttt{Bytes}^{\oplus} \\ \wr X^{\ominus} \int & \triangleq & \texttt{Bytes}^{\ominus} \end{array}$$

Fully-abstract compilation

We consider the extension F_ρ and $\lambda_\rho^{\sigma\oplus/\ominus}$ with integer mutable store of F and $\lambda^{\sigma\oplus/\ominus}$.

Theorem

The polarized compilation scheme $(\!|\cdot|\!)$ from F_ρ to $\lambda_\rho^{\sigma\oplus/\ominus}$ is fully abstract.

- Integer store is needed to prove a key definability result for the trace semantics of F_ρ.
- Implement the freshness test of polarization in $\lambda^{\sigma\oplus/\ominus}$ by storing seals.

Conclusion

A broader setting

- Coinductive reasoning: bisimulations, up-to techniques;
- Presheaves reasoning on resources: Kripke semantics;
- Automated reasoning: symbolic evaluation engine (Higher-Order Constrained Horn Clauses, CAVOC project).

Richer types:

- GADT, Indexed datatypes, disclosable polymorphic references: type constraints in the Type LTS;
- higher-order polymorphism F^{ω}: computing in the Type LTS;
- Dependent types: (intentional) synchronization between the Interactive and the Type LTS?