Modular operational nominal game semantics

Guilhem Jaber (Univ. Nantes)

Scalp Meeting
Deduction System

Computational Model → Environmental Interaction

Computational Model → Deduction System

Environmental Interaction → Deduction System
In this talk

Type System

Operational Semantics

Interactive Semantics

How to define interactive models of programming languages that are:
- abstract with respect to computation
- weakly coupled with the type system

Modularity
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

Type System

Typing judgment
Derivation trees

Operational Semantics

Rewriting systems
\(\lambda\)-calculus

Interactive Semantics

Labelled transition systems
Traces
What is a *good* interactive model?

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

- In a closed world:
  - captured via contextual equivalence;
  - Full abstraction.
How to define the observational power of a programming language?

Via **polarization**:

- *interact* with negative (⊖) values;
- *observe* positive (⊕) values.
Operational Languages

\[ L_{\text{op}} = (\text{Terms}, \rightarrow_{\text{op}}) \text{ with operational reduction relation:} \]

\[ M \rightarrow_{\text{op}} N \]

Example for \( L_{\text{cbv}} \) the pure \( \lambda \)-calculus in call-by-value

Values \( V, W \triangleq x \mid n \mid \lambda x. N \)

Terms \( M, N \triangleq V \mid MN \)

ECxts \( E, E' \triangleq \bullet \mid EM \mid VE \)

\[ E[(\lambda x. M)V] \rightarrow_{\text{op}} E[M\{x := V\}] \]

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

Lemma

A closed term of \( L_{\text{cbv}} \) in normal form is either a value \( V \) or a head normal form \( E[nV] \).
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 $\emptyset \vdash E : \theta \rightsquigarrow \text{Unit}$ and substitution $\emptyset \vdash \gamma : \Gamma \rightarrow \text{Values}$, 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 $L_{\text{cbv}}^{ty}$ that extends $L_{\text{cbv}}$ with:

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

We consider $L_{cbv}^\{ty\}$ that extends $L_{cbv}$ with:

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

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

We consider $L_{cbv}^{ty}$ that extends $L_{cbv}$ with:

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

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

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

\[
\begin{align*}
\text{if } f & \mapsto_\oplus (g, [f \mapsto g]) \\
\text{then } \lambda x. M & \mapsto_\oplus (f, [f \mapsto \lambda x. M]) \\
() & \mapsto_\oplus (((), \varepsilon))
\end{align*}
\]

\[
\begin{align*}
V \mapsto_\oplus (A, \gamma) & \quad W \mapsto_\oplus (B, \gamma') \\
\langle V, W \rangle & \mapsto_\oplus (\langle A, B \rangle, \gamma \cdot \gamma') \\
\text{inj}_i(V) & \mapsto_\oplus (\text{inj}_i(A), \gamma)
\end{align*}
\]

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 blocks;
- inherits a sequential structure and a composition from game semantics;
- provides nominal resource usage control.
The two players exchange *moves*, which are in one of four forms:

<table>
<thead>
<tr>
<th>Move kind</th>
<th>P-question</th>
<th>P-answer</th>
<th>O-question</th>
<th>O-answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move form</td>
<td>$\bar{f}(A)$</td>
<td>$\text{ret}(A)$</td>
<td>$f(A)$</td>
<td>$\text{ret}(A)$</td>
</tr>
</tbody>
</table>

*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 $\mapsto_{\text{op}}$;

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

\[ \mathcal{L}_{\text{ogs}} = (\text{Confs}_{\text{ogs}}, \rightarrow_{\text{ogs}}, a \in \text{Moves} \cup \{\text{op}\}) \]

is the product \( \mathcal{L}_I \times \mathcal{L}_{\text{Ty}} \) of:

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

\[
\begin{array}{cccc}
I \triangleright S & I \xrightarrow{a}_I J & S \xrightarrow{a}_{\text{Ty}} T & J \triangleright T \\
\hline
(I; S) \xrightarrow{a}_{\text{ogs}} (J; T)
\end{array}
\]
Interactive LTS for $L_{\text{cbv}}$

"Abstract machines for Interaction"

$$L_1 = (\text{Confs}_1, \overset{a}{\rightarrow}_1, a \in \text{Moves} \cup \{\text{op}\})$$

- $\overset{\text{op}}{\rightarrow}$ is embedded into $\overset{\text{op}}{\rightarrow}_1$
- visible actions are moves;
- configurations $I$ are either active $\langle M; \sigma; \gamma \rangle$ or passive $\langle \sigma; \gamma \rangle$;
- $\gamma$ is a list of substitutions from names to values;
- $\sigma$ is a stack of evaluation contexts;
The Interactive LTS for $\mathbb{L}_{cbv}$

<table>
<thead>
<tr>
<th>op</th>
<th>$\langle M; \sigma; \gamma \rangle$</th>
<th>$\xrightarrow{\text{op}_{\text{I}}} \langle N; \sigma; \gamma \rangle$ when $M \xrightarrow{\text{op}} N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ</td>
<td>$\langle E[fV]; \sigma; \gamma \rangle$</td>
<td>$\xrightarrow{\text{f(g)}} \langle E :: \sigma; \gamma \cdot [g \mapsto V] \rangle$</td>
</tr>
<tr>
<td>PA</td>
<td>$\langle V; \sigma; \gamma \rangle$</td>
<td>$\xrightarrow{\text{ret}(f)} \langle \sigma; \gamma \cdot [f \mapsto V] \rangle$</td>
</tr>
<tr>
<td>OQ</td>
<td>$\langle \sigma; \gamma \rangle$</td>
<td>$\xrightarrow{\text{f(g)}} \langle \gamma(f)g; \sigma; \gamma \rangle$</td>
</tr>
<tr>
<td>OA</td>
<td>$\langle E :: \sigma; \gamma \rangle$</td>
<td>$\xrightarrow{\text{ret}(f)} \langle E[f]; \sigma; \gamma \rangle$</td>
</tr>
</tbody>
</table>
The Interactive LTS for $L_{cbv}^{ty}$

<table>
<thead>
<tr>
<th>op</th>
<th>$\langle M; \sigma; \gamma \rangle$</th>
<th>$\xrightarrow{\text{op}}_{I}$</th>
<th>$\langle N; \sigma; \gamma \rangle$ when $M \mapsto_{\text{op}} N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ</td>
<td>$\langle E[fV]; \sigma; \gamma \rangle$</td>
<td>$\xrightarrow{\bar{f}(A)}_{I}$</td>
<td>$\langle E :: \sigma; \gamma \cdot \gamma' \rangle$ when $V \mapsto_{\oplus} (A, \gamma')$</td>
</tr>
<tr>
<td>PA</td>
<td>$\langle V; \sigma; \gamma \rangle$</td>
<td>$\xrightarrow{\text{ret}(A)}_{I}$</td>
<td>$\langle \sigma; \gamma \cdot \gamma' \rangle$ when $V \mapsto_{\oplus} (A, \gamma')$</td>
</tr>
<tr>
<td>OQ</td>
<td>$\langle \sigma; \gamma \rangle$</td>
<td>$\xrightarrow{f(A)}_{I}$</td>
<td>$\langle \gamma(f)A; \sigma; \gamma \rangle$</td>
</tr>
<tr>
<td>OA</td>
<td>$\langle E :: \sigma; \gamma \rangle$</td>
<td>$\xrightarrow{\text{ret}(A)}_{I}$</td>
<td>$\langle E[A]; \sigma; \gamma \rangle$</td>
</tr>
</tbody>
</table>
Typing LTS

\[ \mathcal{L}_{Ty} = (\text{Conf}_{Ty}, \overset{a}{\rightarrow}_{Ty}, a \in \text{Moves} \cup \{\text{op}\}) \]

- configurations \( \mathcal{S} \) keep track 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 $L_{cbv}^{ty}$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Transition</th>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>$\langle \Delta_O \vdash \theta; \Sigma; \Delta_P \rangle \xrightarrow{\text{op}} I \langle \Delta_O \vdash \theta; \Sigma; \Delta_P \rangle$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ</td>
<td>$\langle \Delta_O \vdash \theta; \Sigma; \Delta_P \rangle \xrightarrow{f(A)} I \langle \Delta_O \vdash \Sigma'; \Delta_P \cdot \Delta'_P \rangle$ when $\Delta'_P \vdash \oplus A : \theta_1$ with $\Delta_O(f) = \theta_1 \rightarrow \theta_2$ and $\Sigma' = (\theta_2 \leadsto \theta) :: \Sigma$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA</td>
<td>$\langle \Delta_O \vdash \theta; \Sigma; \Delta_P \rangle \xrightarrow{\text{ret}(A)} I \langle \Delta_O \vdash \Sigma; \Delta_P \cdot \Delta'_P \rangle$, when $\Delta'_P \vdash \oplus A : \theta$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OQ</td>
<td>$\langle \Delta_O \vdash \Sigma; \Delta_P \rangle \xrightarrow{f(A)} I \langle \Delta_O \cdot \Delta'_O \vdash \theta'; \Sigma; \Delta_P \rangle$ when $\Delta'_O \vdash \oplus A : \theta$ with $\Delta_P(f) = \theta \rightarrow \theta'$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OA</td>
<td>$\langle \Delta_O \vdash \Sigma; \Delta_P \rangle \xrightarrow{\text{ret}(A)} I \langle \Delta_O \cdot \Delta'_O \vdash \theta'; \Sigma'; \Delta_P \rangle$ when $\Delta'_O \vdash \oplus A : \theta$ and $\Sigma' = (\theta \leadsto \theta') :: \Sigma$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 $\lambda$-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

\[
\begin{align*}
\Gamma, X : \text{Type} & \vdash M : \theta & \Gamma \vdash M : \forall X . \theta \\
\Gamma \vdash \forall X . M : \forall X . \theta & & \Gamma \vdash M \theta : \theta' \{ X := \theta \} \\
\Gamma \vdash M : \theta \{ X := \theta' \} & & \Gamma \vdash \langle \theta'; M \rangle : \exists X . \theta \\
\Gamma \vdash M : \exists X . \theta & \quad \Gamma, X : \text{Type}, x : \theta \vdash N : \theta' & \quad \Gamma \vdash \text{match } M \text{ with } (X, x) \Rightarrow N : \theta'
\end{align*}
\]
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 \( \text{pol}_\eta(\cdot)^\kappa : \text{Types} \to \text{Types} \), for \( \kappa \in \{\oplus, \ominus\} \), defined as

\[
\begin{align*}
\text{pol}_\eta(\theta \to \theta')^\kappa & \triangleq \text{pol}_\eta(\theta)^\kappa \to \text{pol}_\eta(\theta')^\kappa \\
\text{pol}_\eta(\theta \times \theta')^\kappa & \triangleq \text{pol}_\eta(\theta)^\kappa \times \text{pol}_\eta(\theta')^\kappa \\
\text{pol}_\eta(\theta + \theta')^\kappa & \triangleq \text{pol}_\eta(\theta)^\kappa + \text{pol}_\eta(\theta')^\kappa \\
\text{pol}_\eta(\forall X.\theta)^\kappa & \triangleq \forall X.\text{pol}_\eta.[X \mapsto \kappa^\perp]{\theta^\kappa} \\
\text{pol}_\eta(\exists X.\theta)^\kappa & \triangleq \exists X.\text{pol}_\eta.[X \mapsto \kappa]{\theta^\kappa} \\
\text{pol}_\eta(X)^\kappa & \triangleq \eta(X)
\end{align*}
\]
Positive Typing for Polymorphism

- Typing judgment \( \Gamma^X \) indexed by player \( X \in \{P, O\} \).
- Splitting typing contexts \( \Delta_O|\Delta_P \).

\[
\Delta_O(p) = \alpha \\
\Delta_O|\Delta_P \vdash^P p : \alpha^\oplus
\]

\[
\Delta_P(p) = \alpha \\
\Delta_O|\Delta_P \vdash^O p : \alpha^\ominus
\]

\[
p \in \text{PNames} \setminus \text{dom}(\Delta_O \cdot \Delta_P) \\
\Delta_O|\Delta_P \vdash^P \nu p : \alpha^\oplus
\]

\[
p \in \text{PNames} \setminus \text{dom}(\Delta_O \cdot \Delta_P) \\
\Delta_O|\Delta_P \vdash^O \nu p : \alpha^\ominus
\]

\[
\Delta_O|\Delta_P, \alpha : \text{Type} \vdash^P A : \theta'\{X := \alpha\} \\
\Delta_O|\Delta_P \vdash^P \langle \nu \alpha; A \rangle : \exists X.\theta'
\]

\[
\Delta_O, \alpha : \text{Type}|\Delta_P \vdash^O A : \theta'\{X := \alpha\} \\
\Delta_O|\Delta_P \vdash^O \langle \nu \alpha; A \rangle : \exists X.\theta'
\]
Memoryful programs

\( \mathcal{L}_{op} = (\text{Terms, Stores, } \rightarrow_{op}) \) with \textit{operational reduction relation}:

\[(M, \xi) \rightarrow_{op} (N, \zeta)\]

Example for the \( \nu \)-calculus, with dynamic creation of \textit{atoms} \( an \) stored in the set \( S \):

\[
\begin{align*}
(E[(\lambda x. M) V], S) &\quad \rightarrow_{op} \quad (E[M\{x := V\}], S) \\
(E[\text{if true then } N_1 \text{ else } N_2], S) &\quad \rightarrow_{op} \quad (N_1, S) \\
(E[\text{if false then } N_1 \text{ else } N_2], S) &\quad \rightarrow_{op} \quad (N_2, S) \\
(E[a = a], S) &\quad \rightarrow_{op} \quad (E[\text{true}], S) \\
(E[a = a'], S) &\quad \rightarrow_{op} \quad (E[\text{false}], S) \\
(E[\text{new } x \text{ in } M], S) &\quad \rightarrow_{op} \quad (E[M\{x := a\}], S \uplus \{a\})
\end{align*}
\]
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

\[ \mathcal{L}_H = (\text{Confs}_H, \xrightarrow{a}_H, a \in \text{Moves} \cup \{\text{op}\}) \]

- 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.

- $\langle M; S; \sigma; \gamma \rangle \vdash \langle D \rangle \xrightarrow{\text{op}} \langle N; S'; \sigma; \gamma \rangle \vdash \langle D \rangle$
- when $(M, S) \leftrightarrow_{\text{op}} (N, S')$

- $\langle E[xV]; S; \sigma; \gamma \rangle \vdash \langle D \rangle \xrightarrow{\overline{f}(A)} \langle S; E :: \sigma; \gamma \cdot \gamma' \rangle \vdash \langle D \cup D' \rangle$
- when $V \mapsto_{\oplus} (A, \gamma')$

- $\langle V; S; \sigma; \gamma \rangle \vdash \langle D \rangle \xrightarrow{\text{ret}(A)} \langle S; \sigma; \gamma \cdot \gamma' \rangle \vdash \langle D \cup D' \rangle$
- when $V \mapsto_{\oplus} (A, \gamma')$

- $\langle S; \sigma; \gamma \rangle \vdash \langle D \rangle \xrightarrow{f(A)} \langle VA; S \cup D'; \sigma; \gamma \rangle \vdash \langle D \cup D' \rangle$
- when $\gamma(f) = V$

- $\langle S; E :: \sigma; \gamma \rangle \vdash \langle D \rangle \xrightarrow{\text{ret}(A)} \langle E[A]; S \cup D'; \sigma; \gamma \rangle \vdash \langle D \cup D' \rangle$

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

$$D' \cap S = \emptyset$$
1 Proving Full-Abstraction

2 Sumii & Pierce compilation for polymorphism
Composition/Interaction operational reduction

- **a CIO reduction relation** $\xrightarrow{\text{cio}}$ extend $\rightarrow_1$ into parallel composition plus hiding.
- **configurations** $\mathcal{D}, \mathcal{D}' \in \text{Confs}_{\text{cio}} \triangleq \{ \nu(G||H) \mid G \perp H \}$
- $\xrightarrow{\text{cio}}$ defined by the following rules:

$$
\begin{align*}
G \xrightarrow{m} G' & \quad \text{and} \quad H \xrightarrow{m^\perp} H' \\
\nu(G||H) \xrightarrow{\text{cio}} \nu(G'||H')
\end{align*}
$$
Adequacy via abstract machines

Introduce an operational reduction reduction

\[(\text{Confs}_{\text{bop}}, \mapsto_{\text{bop}})\]

together with two functional bisimulation \(\kappa_{\text{bop}} : \text{Confs}_{\text{bop}} \to \text{Terms} \times \text{Stores}\) and \(\kappa_{\text{cio}} : \text{Confs}_{\text{cio}} \to \text{Confs}_{\text{bop}}\).

In \(\text{L}_{\text{cbv}}\):

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

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

\[
V \mapsto_{\oplus} (A, \delta') \\
(V; E :: \sigma; \delta) \mapsto_{\text{bop}} (E[A]; \sigma; \delta \cdot \delta')
\]

\[
M \mapsto_{\text{op}} N \\
(M; \sigma; \delta) \mapsto_{\text{bop}} (N; \sigma; \delta)
\]
## Soundness of trace equivalence

Suppose we observe only booleans.

### Definition

We write \((G, H) \in \perp \perp\) when there exists a complete traces \(t\) such that \(t \in CTr(G)\) and \(t \perp \cdot \text{ret}(b) \in CTr(H)\) with \(b \in \{\text{true}, \text{false}\}\).

### Lemma

\((G, H) \in \perp \perp\) if and only if \(\nu(G \| H) \Downarrow\).

### Lemma

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

### 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 $\nu(\Gamma \vdash M : \theta) \simeq_{ctr} \nu(\Gamma \vdash N : \theta)$.

- 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**

\(|\cdot| : L_1 \rightarrow L_2\) is fully abstract if for all terms \(M, N\) of \(L_1\), we have

\(M \sim_{L_1}^\text{ctx} N\) iff \(|M| \sim_{L_2}^\text{ctx} |N|\).

Suppose \(L_2\)-contexts are more powerful than \(L_1\)-contexts:

- \(\leftrightarrow^2\) is contained in \(\leftrightarrow^1\): \(L_2\)-contexts can observe more.
- \(\vdash^1\) is contained in \(\vdash^2\): \(L_2\)-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 \(\leftrightarrow^1\) does but not \(\leftrightarrow^2\);
- reject on Opponent interactions the positive values validated by \(\vdash^2\) but not by \(\vdash^1\).
Proving full-abstraction results for compilers

**Theorem**

\[
\langle \cdot \rangle : L_1 \rightarrow L_2 \text{ is fully abstract when:}
\]

- \(\sim_{tr}^{L_1}\) is fully abstract wrt \(\sim_{ctx}^{L_1}\)
- \(\sim_{tr}^{L_2}\) is sound wrt \(\sim_{ctx}^{L_2}\)
- \(\langle \cdot \rangle\) induces a bisimulation between \(L_{OGS}(L_1)\) and \(L_{OGS}(L_2)\).
1 Proving Full-Abstraction

2 Sumii & Pierce compilation for polymorphism
In 2000, Sumii & Pierce proposed a compilation scheme \( (\cdot) : F \to L_c \) between:

- the (second-order) polymorphic \( \lambda \)-calculus \( F \);
- the cryptographic \( \lambda \)-calculus \( L_c \), a simply-typed \( \lambda \)-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 $\lambda$-calculus

- Sealed values $\{V\}_\sigma$ with a seal $\sigma$
- Dynamic seal creation `newseal $\sigma$ in $M$`
- Unsealing:
  
  $$\text{match } \{V\}_\sigma \text{ with } (\sigma', x) \Rightarrow M | wrong \Rightarrow N \Rightarrow_{\text{op}} \begin{cases} M\{x := V\} \text{ when } \sigma = \sigma' \\ N \text{ otherwise} \end{cases}$$

- Type $\text{Seal}_\theta$ for seals that can be used only on values of type $\theta$
  $\rightsquigarrow$ needed to ensure type soundness.
Embedding $F$ into $\mathbb{L}_c$

Type Erasure $\exists \cdot \int : F \rightarrow \mathbb{L}_c$

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

\[
\begin{align*}
\exists X.\theta &\triangleq Unit \rightarrow \int \theta \\
\forall X.\theta &\triangleq Unit \times \int \theta \\
\theta \rightarrow \theta' &\triangleq \int \theta \rightarrow \int \theta' \\
X^{\oplus} &\triangleq \text{Bytes} \\
X^{\ominus} &\triangleq \text{Bytes}
\end{align*}
\]
\( (\cdot) : F \rightarrow L_c \) embeds some runtime checks at the interaction points:

- for Player transition:
  - at type \( X^{\oplus} \): it seals using \( \sigma_X \) the value exchanged
  - at type \( X^{\ominus} \): it does nothing.

- for Opponent transition:
  - at type \( X^{\oplus} \): it unseals using \( \sigma_X \) the value provided
  - at type \( X^{\ominus} \): it does nothing.
Sumii & Pierce compilation scheme (II/III)

\[
\begin{align*}
\text{protect}_{\eta, \text{Unit}}^\text{Seal} x & \triangleq x \\
\text{protect}_{\eta, \theta_1 \times \theta_2}^\text{Seal} x & \triangleq \text{let } x_1 = \pi_1(x) \text{ in let } x_2 = \pi_2(x) \text{ in } \langle \text{protect}_{\eta, \theta_1}^\text{Seal} x_1, \text{protect}_{\eta, \theta_2}^\text{Seal} x_2 \rangle \\
\text{protect}_{\eta, \theta \rightarrow \theta'}^\text{Seal} x & \triangleq \lambda y. \text{let } z = x(\text{confine}_{\eta, \theta}^\text{Seal} y) \text{ in } \text{protect}_{\eta, \theta'}^\text{Seal} z \\
\text{protect}_{\eta, \forall X. \theta}^\text{Seal} x & \triangleq \lambda. \text{let } y = x() \text{ in } \text{protect}_{\eta, \theta}^\text{Seal} x \{ X := \alpha^- \} \\
\text{protect}_{\eta, \exists X. \theta}^\text{Seal} x & \triangleq \text{let } y = \pi_2(x) \text{ in } \nu \alpha.\langle () \rangle, \text{protect}_{\eta \oplus \{ \alpha \}, \theta}^\text{Seal} x \{ X := \alpha^+ \} y \\
\text{protect}_{\eta, \alpha \oplus}^\text{Seal} x & \triangleq \text{seal}_\alpha x \\
\text{protect}_{\eta, \alpha \ominus}^\text{Seal} x & \triangleq x
\end{align*}
\]
Sumii & Pierce compilation scheme (III/III)

$\text{confine}^{Seal}_{\eta, \text{Unit}} x \triangleq x$

$\text{confine}^{Seal}_{\eta, \theta_1 \times \theta_2} x \triangleq \text{let } x_1 = \pi_1(x) \text{ in let } x_2 = \pi_2(x) \text{ in } \langle \text{confine}^{Seal}_{\eta, \theta_1} x_1, \text{confine}^{Seal}_{\eta, \theta_2} x_2 \rangle$

$\text{confine}^{Seal}_{\eta, \theta \rightarrow \theta'} x \triangleq \lambda y. \text{let } z = x(\text{protect}^{Seal}_{\eta, \theta} y) \text{ in confine}^{Seal}_{\eta, \theta'} z$

$\text{confine}^{Seal}_{\eta, \forall X. \theta} x \triangleq \lambda_. \text{let } y = x() \text{ in } \nu \alpha. \text{confine}^{Seal}_{\eta \cup \{\alpha\}, \theta\{X:=\alpha^+\}} y$

$\text{confine}^{Seal}_{\eta, \exists X. \theta} x \triangleq \text{let } y = \pi_2(x) \text{ in } \langle(), \text{confine}^{Seal}_{\eta, \theta\{X:=\alpha^-\}} y\rangle$

$\text{confine}^{Seal}_{\eta, \alpha^\oplus} x \triangleq \text{unseal}_\alpha x$

$\text{confine}^{Seal}_{\eta, \alpha^\ominus} x \triangleq x$
In 2018, Devriese, Patrignani & Piessens provide a counterexample to the fully-abstract conjecture based on the universal type:

$$\exists Y. \forall X. (X^\oplus \to Y^\ltimes) \times (Y^\ltimes \to X^\oplus)$$

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

\(\langle \cdot \rangle : F \rightarrow L_c\) embeds some runtime checks at the interaction points:

- for Player transition:
  - at type \(X^\oplus\): it seals using \(\sigma_X\) the value exchanged
  - at type \(X^\ominus\): it does nothing.

- for Opponent transition:
  - at type \(X^\oplus\): it unseals using \(\sigma_X\) the value provided
  - at type \(X^\ominus\): it does nothing.
Fully-abstract compilation

**Theorem**

\[ (\cdot) : F \rightarrow L_c \text{ is fully abstract if for all terms } M, N \text{ of } F, \text{ we have } M \simeq_{tr}^F N \iff (M) \simeq_{tr}^{L_c} (N). \]

\( L_c \)-contexts are more powerful than \( F \)-contexts:

- \( \leftrightarrow^{L_c} \) is embedded in \( \leftrightarrow^F \): \( F \)-contexts can observe more.
- \( \vdash^F \) is embedded in \( \vdash^{L_c} \): \( L_c \)-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 \( \leftrightarrow^F \) does but not \( \leftrightarrow^{L_c} \)

- reject on Opponent interactions the abstract values validated by \( \vdash^{L_c} \) but not by \( \vdash^F \).
(\cdot) : F \rightarrow L_c embeds some runtime checks at the interaction points:

- for Player transition:
  - at type \( \mathcal{X}^{\oplus} \): it seals using \( \sigma_{\mathcal{X}} \) the value exchanged
  - at type \( \mathcal{X}^{\ominus} \): it does nothing.

- for Opponent transition:
  - at type \( \mathcal{X}^{\oplus} \): it unseals using \( \sigma_{\mathcal{X}} \) the value provided
  - at type \( \mathcal{X}^{\ominus} \): it checks for freshness of the value provided.
Polarizing the cryptographic \(\lambda\)-calculus

- \(\lambda^{\sigma\oplus/\ominus}\): Add polarization annotations \(\text{Bytes}^{\oplus}, \text{Bytes}^{\ominus}\) to \(L_c\)
  - \(\leadsto\) positive when we can unseal the value;
  - \(\not\leadsto\) negative when we cannot.

- Polarization may evolve from \(\oplus\) to \(\ominus\) during the interaction!

- Use polarity information to define \(\vdash^{\lambda^{\sigma\oplus/\ominus}}\)

- Type Erasure \(\Jane\cdot\int: F \rightarrow \lambda^{\sigma\oplus/\ominus}\):

\[
\begin{align*}
\Jane X^{\oplus} & \triangleq \text{Bytes}^{\oplus} \\
\Jane X^{\ominus} & \triangleq \text{Bytes}^{\ominus}
\end{align*}
\]
Fully-abstract compilation

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

**Theorem**

The polarized compilation scheme $\|\|$ from $F_\rho$ to $\lambda^{\oplus/\ominus}_\rho$ is fully abstract.

- Integer store is needed to prove a key definability result for the trace semantics of $F_\rho$.
- Implement the freshness test of polarization in $\lambda^{\oplus/\ominus}_\sigma$ 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 $F^\omega$: computing in the Type LTS;
- Dependent types: (intentional) synchronization between the Interactive and the Type LTS?