Abstract machines for game semantics, revisited

Olle Fredriksson    Dan R. Ghica
University of Birmingham

Friday, June 28 – LICS 2013

[arXiv:1304.4159 [cs.LO]]
Platform independence

- Write a program once – deploy to several different platforms.

✓ Programming languages with cross-platform compilers and libraries.
Platform independence

- Write a program once – deploy to several different platforms.
  ✓ Programming languages with cross-platform compilers and libraries.

Architecture independence

- Write a program once – deploy to several different architectures.
  ?

Not solved by domain-specific languages or libraries!
Seamless computing

is the compilation of conventional programs to unconventional targets.

Distributed systems, hardware circuits, …
Previously...
**Geometry of Interaction**

- Originally a semantics for linear logic.
- Based on the idea of *computation as interaction*.
- Operational models based on token-passing abstract machines.
Distribution using Geometry of Interaction

\[
\text{let } f = \lambda x. x \ast x \text{ in } f \ 3 + f \ 4
\]

Distribution using Geometry of Interaction

\[ \text{(let } f = (\lambda x. x \times x)@B \text{ in } f 3 + f 4)@A \]
Distribution using Geometry of Interaction

Shortcomings:
- Inherently sequential?\(^1\) How to do language constructs like `par, newvar, . . .`?
- Tokens grow large.

Today's experiment:

- Game models have operational content.
- We have game models for concurrency.

GoI $\rightarrow$ Token machines
Games $\rightarrow$ ???

Idea: Construct machines that have plays as operational traces.

--

A SHORT DIGRESSION

A play is a sequence of moves, e.g.:

$$q_1 \cdot q_2 \cdot a_2 \cdot a_1$$

Tells us a possible interaction with a program.
But wait, there’s more!

Consider:

\[ \lambda f. f (\lambda x. f (\lambda y. x)) \quad \lambda f. f (\lambda x. f (\lambda y. y)) \]
But wait, there’s more!

\[ \lambda f. f \ (\lambda x. f \ (\lambda y. x)) \]

\[ \lambda f. f \ (\lambda x. f \ (\lambda y. y)) \]

\[ ((\iota_1 \to \iota_2) \to \iota_3) \to \iota_4) \]
But wait, there’s more!

\[
\lambda f. f (\lambda x. f (\lambda y. x))
\]

\[
((\iota_1 \to \iota_2) \to \iota_3) \to \iota_4
\]

\[
\lambda f. f (\lambda x. f (\lambda y. y))
\]

\[
((\iota_1 \to \iota_2) \to \iota_3) \to \iota_4
\]
How to define the concatenation of two plays? Reindexing is necessary.
The moves carry the justification structure using *names*:

\[
q_1 p_0[p_1] \cdot q_2 p_1[p_2] \cdot a_2 p_2 \cdot a_1 p_1
\]

Leads to:

- Simplified proofs.
- Self-contained traces.

Operational content “restored”!

---

Today’s experiment:

GoI $\rightarrow$ Token machines
Games $\rightarrow$ ???

(Revised) idea: Construct machines that have plays in the nominal model as operational traces.
HRAMs and nets
A “realistic” model of computers.
HRAMs

Heap and Register Abstract Machines.
HRAMs

Instructions

\[
\text{Instr} ::= i \leftarrow \text{new } j, k \\
| i, j \leftarrow \text{get } k \\
| \text{update } i, j \\
| \text{free } i \\
| \text{flip } i, j \\
| i \leftarrow \text{set } j \\
| \text{ifzero } i c_1 c_2
\]
If $E = (A, P)$, then

1. $(\text{spark } a, \bar{d}) \xrightarrow{(\chi(a), \bar{d})} _{E, \chi} \emptyset$, \quad (O, \chi(a)) \notin A
2. $(\text{spark } a, \bar{d}) \xrightarrow{} _{E, \chi} (P(\chi(a)), \bar{d})$, \quad (O, \chi(a)) \in A
3. $\emptyset \xrightarrow{(a, \bar{d})^*} _{E, \chi} (P(a), \bar{d})$, \quad (O, a) \in A

Enables seamlessness!
HRAM nets

CHAM-style operational semantics.
**Denotational semantics**

\[ [S] \triangleq \{ s \mid \exists n.\text{initial}(S) \overset{s}{\rightarrow} n \} \]

\[ [S_1; S_2] = [S_1]; [S_2] \]
\[ \triangleq \{ s|B \mid s \in \text{traces}_{A \otimes B \otimes C} \land s|C \in [S_1] \land \pi \cdot s^B|A \in [S_2] \} \]
\[ = \text{Synchronisation and hiding} \]

\[ [S_1 \otimes S_2] = [S_1] \otimes [S_2] \]
\[ \triangleq \{ s \mid s \in \text{traces}_{A \otimes B} \land s|B \in [S_1] \land s|A \in [S_2] \} \]
\[ = \text{Trace interleaving} \]

**Theorem**

*HRAM nets form a symmetric compact-closed category, called HRAMnet.*
Game nets
HRAM nets whose denotation corresponds to games.

Idea:
Ports ↔ moves
Heap pointers ↔ justification pointers
**Plug and Play?**

Invariant: Each (non-initial) message has one known and one unknown pointer.
**Plug and Play?**

Invariant: Each (non-initial) message has one known and one unknown pointer.

The interpretation of $x : int \vdash S(S(x)) : int$ in a context $C[-int] : int$:

\[
\begin{array}{c}
(p_2, p_3) \\
S \quad (p_1, p_2) \\
S \quad (p_0, p_1) \\
\vdash \\
C[-] \\
\end{array}
\]

\[
\begin{array}{c}
\text{int}_1 \Rightarrow \text{int}_2 \\
q_2^O \\
\text{q}_1^P \\
\text{q}_1^O \\
\text{2}^P \\
\end{array}
\]
**PLUG AND PLAY?**

Invariant: Each (non-initial) message has one known and one unknown pointer.

The interpretation of $x : \text{int} \vdash S(S(x)) : \text{int}$ in a context $C[\neg \text{int}] : \text{int}$:

NO! The invariant breaks.
Composition must be mediated.
HRAMs for Copycat

\[ \alpha : (\text{com}_1 \Rightarrow \text{com}_2) \rightarrow (\text{com}_3 \Rightarrow \text{com}_4) \]

Nominally:

\[ r_{4p_0[p_1]} \cdot r_{2p_1[p_2]} \cdot r_{1p_2[p_3]} \cdot r_{3p_1[p_4]} \cdot d_{3p_4} \cdots \]
HRAMs for Copycat

\[ \alpha : (\text{com}_1 \Rightarrow \text{com}_2) \Rightarrow (\text{com}_3 \Rightarrow \text{com}_4) \]

\[ r_2 : [p_2 \mapsto p_1] \]
\[ r_3 : [p_2 \mapsto p_1, p_4 \mapsto p_3] \]
\[ d_1 : [p_2 \mapsto p_1] \]
\[ d_4 : [] \]

Nominally:

\[ r_4 p_0[p_1] \cdot r_2 p_1[p_2] \cdot r_1 p_2[p_3] \cdot r_3 p_1[p_4] \cdot d_3 p_4 \cdot \ldots \]
\[
\begin{align*}
cci & \triangleq \text{flip } 0, 1; 1 \leftarrow \text{new } 0, 3 \\
ccq & \triangleq 1 \leftarrow \text{new } 1, 3; 0, 3 \leftarrow \text{get } 0 \\
cca & \triangleq \text{flip } 0, 1; 0, 3 \leftarrow \text{get } 1; \text{free } 1
\end{align*}
\]

For game interfaces \( \mathcal{A} \) and \( \mathcal{A}' \) such that \( \pi \vdash \mathcal{A} =_A \mathcal{A}' \), we define a generalised copycat engine as \( \mathcal{C}_{C, \pi, \mathcal{A}} = (A \Rightarrow A', P) \), where:

\[
P \triangleq \{ q_2 \mapsto C; \ \text{spark } q_1 \mid q_2 \text{ initial in } A' \} \\
\cup \{ q_2 \mapsto ccq; \ \text{spark } q_1 \mid q_2 \text{ non-initial question in } A' \} \\
\cup \{ a_2 \mapsto cca; \ \text{spark } a_1 \mid a_2 \text{ answer in } A' \} \\
\cup \{ q_1 \mapsto ccq; \ \text{spark } q_2 \mid q_1 \text{ question in } A \} \\
\cup \{ a_1 \mapsto cca; \ \text{spark } a_2 \mid a_1 \text{ answer in } A \}
\]
GAME COMPOSITION

\((A \Rightarrow B) \otimes (B' \Rightarrow C) \rightarrow (A' \Rightarrow C')\)

\(q_6^O\)

\(q_4^P\)

\(q_3^O\)

\(q_2^P\)

\(q_1^O\)

\(q_5^P\)

\(q_6^O p_0[p_1] \cdot q_4^P p_1[p_2] \cdot q_3^O p_2[p_3] \cdot q_2^P p_1[p_4] \cdot q_1^O p_4[p_5] \cdot q_5^P p_1[p_6] \cdot \cdots\)
GAME COMPOSITION

\[(A \Rightarrow B) \otimes (B' \Rightarrow C) \rightarrow (A' \Rightarrow C')\]

\[q_6^O \rightarrow cci \rightarrow q_6^P\]

\[q_4^P \rightarrow cciq \rightarrow q_3^O\]

\[q_2^P \rightarrow ccq \rightarrow q_1^O\]

\[q_5^P \rightarrow cciq \rightarrow q_4^O\]

\[q_6p_0[p_1] \cdot q_4p_1[p_2] \cdot q_3p_2[p_3] \cdot q_2p_1[p_4] \cdot q_1p_4[p_5] \cdot q_5p_1[p_6] \cdot \cdots\]
**Game Composition**

\[(A \Rightarrow B) \otimes (B' \Rightarrow C) \rightarrow (A' \Rightarrow C')\]

\[q_0 \Rightarrow \quad \otimes \quad \Rightarrow \quad \rightarrow \quad \Rightarrow \quad \Rightarrow \]

\[q_6^O \quad q_4^P \quad q_3^O \quad q_2^O \quad q_1^O \quad q_5^P \]

\[q_6p_0[p_1] \cdot q_4p_1[p_2] \cdot q_3p_2[p_3] \cdot q_2p_1[p_4] \cdot q_1p_4[p_5] \cdot q_5p_1[p_6] \cdots\]
Game composition

\[(A \Rightarrow B) \otimes (B' \Rightarrow C) \rightarrow (A' \Rightarrow C')\]

\[q_6^O \rightarrow q_4^O \rightarrow q_2^O \rightarrow q_1^O \rightarrow q_5^P \rightarrow q_4^P \rightarrow q_6^P\]

\[q_6 p_0[p_1] \cdot q_4 p_1[p_2] \cdot q_3 p_2[p_3] \cdot q_2 p_1[p_4] \cdot q_1 p_4[p_5] \cdot q_5 p_1[p_6] \cdots\]
GAME COMPOSITION

\[(A \Rightarrow B) \otimes (B' \Rightarrow C) \rightarrow (A' \Rightarrow C')\]

\[q_6^O \quad q_4^P \quad q_3^O \quad q_2^P \quad q_1^O \quad q_5^P\]

\[q_6 p_0[p_1] \cdot q_4 p_1[p_2] \cdot q_3 p_2[p_3] \cdot q_2 p_1[p_4] \cdot q_1 p_4[p_5] \cdot q_5 p_1[p_6] \cdots\]
GAME COMPOSITION
GAME COMPOSITION

Let $A'$, $B'$, and $C'$ be game interfaces such that $\pi_A \vdash A =_A A'$, $\pi_B \vdash B =_A B'$, $\pi_C \vdash C =_A C'$, and

$$(A' \Rightarrow A, P_A) = C \exq,\pi_A^{-1},A'$$

$$(B \Rightarrow B', P_B) = C \exi,\pi_B,B$$

$$(C \Rightarrow C', P_C) = C \cci,\pi_C,C, \text{ where}$$

$$\text{exi} \triangleq 0, 3 \leftarrow \text{get} 0; 1 \leftarrow \text{new} 1, 0$$

$$\text{exq} \triangleq \emptyset, 0 \leftarrow \text{get} 0; 1 \leftarrow \text{new} 1, 3$$

Then the game composition operator $K_{A,B,C}$ is:

$$K_{A,B,C} \triangleq ((A \Rightarrow B) \otimes (B' \Rightarrow C) \Rightarrow (A' \Rightarrow C'), P_A \cup P_B \cup P_C).$$
\textbf{Newvar}

\[ \text{newvar} \triangleq ((\exp_1 \otimes (\exp_2 \Rightarrow \text{com}_3) \Rightarrow \exp_4) \Rightarrow \exp_5, P) \]

\[ P \triangleq \{ q_5 \mapsto 3 \leftarrow \text{set} \, 0; \, \text{cci}; \, \text{spark} \, q_4, \]

\[ q_1 \mapsto \emptyset, 2 \leftarrow \text{get} \, 0; \, \text{flip} \, 0, 1; \, 1 \leftarrow \text{set} \, \emptyset; \]

\[ \text{spark} \, a_1, \]

\[ q_3 \mapsto \text{flip} \, 0, 1; \, 1 \leftarrow \text{new} \, 0, 1; \, \text{spark} \, q_2, \]

\[ a_2 \mapsto \emptyset, 3 \leftarrow \text{get} \, 0; \, \text{update} \, 3 \, 2; \, \text{cca}; \, \text{spark} \, a_3, \]

\[ a_4 \mapsto \text{cca}; \, \text{spark} \, a_5 \} \]

\textit{par, fix, ...}
Demo
Correctness

Theorem

*The compilation of Idealised Concurrent Algol to HRAMs is sound.*
Summary

- Nominal games (for ICA) = specification.
- HRAMs that implement specification.
- Compiler for HRAMs targeting message-passing networks.
THE GOOD AND THE BAD

Benefits:
▶ Seamless distribution of higher-order concurrent PL with state.
▶ No garbage collection required.
▶ Nominal games gives us:
  ▶ No pointer encoding/decoding. Exploits the HRAM ability to create names.
  ▶ Constant token size – the computation history is stored in HRAM heaps rather than the token (cf. IAM, GoI).

Further work:
▶ Inefficiencies:
  ▶ Pointer manipulation from composition and diagonals.
  ▶ Overhead: A large constant.
  ▶ Can we use conventional compilation techniques locally?

Compiler and source available at veritygos.org/gams