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

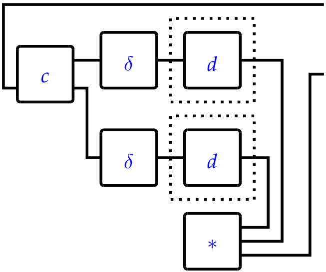
is the compilation of conventional programs to unconventional targets.

Distributed systems, hardware circuits, . . .

Previously…

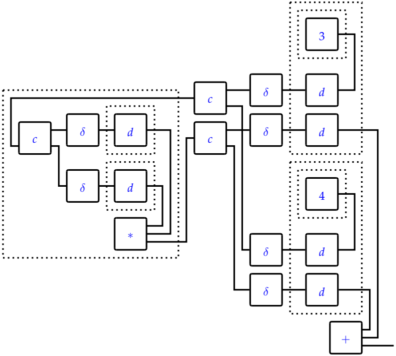
Geometry of Interaction

- 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

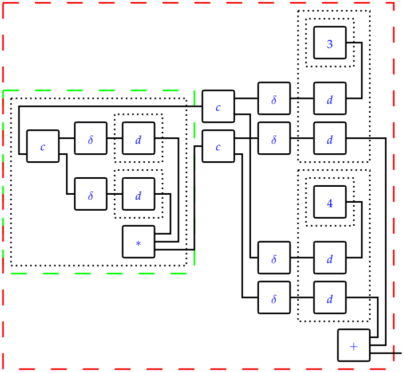
$$let\, f = \lambda x.\, x * x\; in\, f\; 3 + f\; 4$$

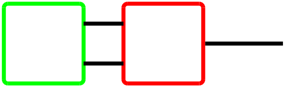


O. Fredriksson and D. R. Ghica. Seamless distributed computing from the geometry of interaction. *Trustworthy Global Computing*, 2012. `veritygos.org`

Distribution using Geometry of Interaction

$$(let\ f = (\lambda x.\ x * x)@B\ in\ f\ 3 + f\ 4)@A$$





O. Fredriksson and D. R. Ghica. Seamless distributed computing from the geometry of interaction. *Trustworthy Global Computing*, 2012. `veritygos.org`

Distribution using Geometry of Interaction

Shortcomings:

- Inherently sequential?[1] How to do language constructs like *par*, *newvar*, ...?
- Tokens grow large.

[1] Olivier Laurent. Étude de la polarisation en logique. Thèse de Doctorat, 2002.

Today's experiment:

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

$$\begin{aligned} \text{GoI} &\longrightarrow \text{Token machines} \\ \text{Games} &\longrightarrow \text{???}^{2} \end{aligned}$$

Idea: Construct machines that have plays as operational traces.

[2] V. Danos, H. Herbelin, and L. Regnier. Game Semantics & Abstract Machines. *LICS*, 1996.

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:

$$\lambda f.\, f\ (\lambda x.\, f\ (\lambda y.\, x)) \qquad\qquad \lambda f.\, f\ (\lambda x.\, f\ (\lambda y.\, y))$$

But wait, there's more!

$$\lambda f.\, f\ (\lambda x.\, f\ (\lambda y.\, x))$$

$$((\iota_1 \rightarrow \iota_2) \rightarrow \iota_3) \rightarrow \iota_4)$$

$$q_4^{\mathbf{O}}$$
$$q_3^{\mathbf{P}}$$
$$q_2^{\mathbf{O}}$$
$$q_3^{\mathbf{P}}$$
$$q_2^{\mathbf{O}}$$
$$q_1^{\mathbf{P}}$$
$$a_1^{\mathbf{O}}$$

$$\lambda f.\, f\ (\lambda x.\, f\ (\lambda y.\, y))$$

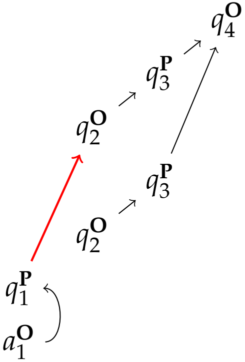
$$((\iota_1 \rightarrow \iota_2) \rightarrow \iota_3) \rightarrow \iota_4)$$

$$q_4^{\mathbf{O}}$$
$$q_3^{\mathbf{P}}$$
$$q_2^{\mathbf{O}}$$
$$q_3^{\mathbf{P}}$$
$$q_2^{\mathbf{O}}$$
$$q_1^{\mathbf{P}}$$
$$a_1^{\mathbf{O}}$$

But wait, there's more!

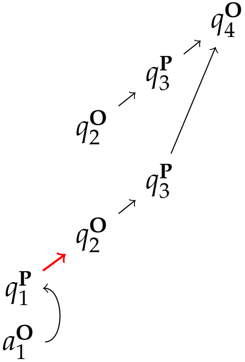
$$\lambda f.\, f\ (\lambda x.\, f\ (\lambda y.\, x))$$

$$((\iota_1 \to \iota_2) \to \iota_3) \to \iota_4)$$



$$\lambda f.\, f\ (\lambda x.\, f\ (\lambda y.\, y))$$

$$((\iota_1 \to \iota_2) \to \iota_3) \to \iota_4)$$



Hyland-Ong games

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

$$+$$

$$0 \cdot 1 \cdot 2 \cdot 1$$

$$=$$

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

How to define the concatenation of two plays? Reindexing is necessary.

J.M.E. Hyland and C.-H.L. Ong. On Full Abstraction for PCF: I, II, and III. *Inf. Comput.*, 2000.

Game semantics in the nominal model

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”!

M. Gabbay and D. R. Ghica. Game semantics in the nominal model. *Electr. Notes Theor. Comput. Sci.*, 2012.

Today's experiment:

$$\begin{aligned} \text{GoI} &\longrightarrow \text{Token machines} \\ \text{Games} &\longrightarrow \text{???} \end{aligned}$$

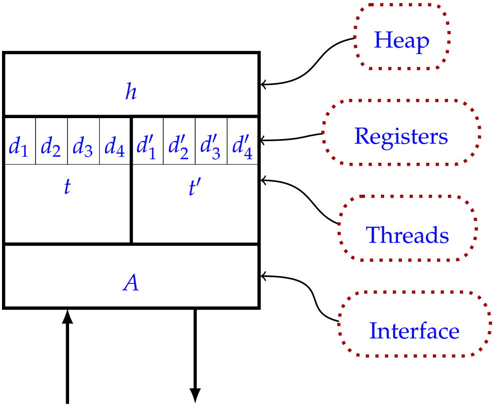
(Revised) idea: Construct machines that have plays *in the nominal model* as operational traces.

HRAMs and nets

A “realistic” model of computers.

HRAMs

Heap and **R**egister **A**bstract **M**achines.



HRAMs

$$
\begin{array}{lll}
\text{Instructions} & \text{Instr} & ::= i \leftarrow \texttt{new}\, j, k \\
 & & \mid i, j \leftarrow \texttt{get}\, k \\
 & & \mid \texttt{update}\, i, j \\
 & & \mid \texttt{free}\, i \\
 & & \mid \texttt{flip}\, i, j \\
 & & \mid i \leftarrow \texttt{set}\, j \\
 & & \mid \texttt{ifzero}\, i\, c_1\, c_2
\end{array}
$$

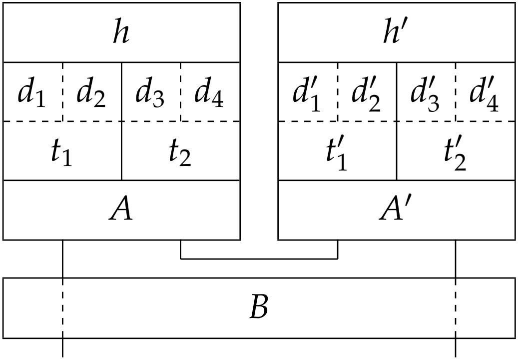
HRAM SEMANTICS

If $E = (A, P)$, then

$$
\begin{aligned}
(\texttt{spark}\, a, \overline{d}) &\xrightarrow[E,\chi]{(\chi(a),\overline{d})} \emptyset, && (\mathbf{O}, \chi(a)) \notin A \\
(\texttt{spark}\, a, \overline{d}) &\xrightarrow[E,\chi]{} (P(\chi(a)), \overline{d}), && (\mathbf{O}, \chi(a)) \in A \\
\emptyset &\xrightarrow[E,\chi]{(a,\overline{d})^{\bullet}} (P(a), \overline{d}), && (\mathbf{O}, a) \in A
\end{aligned}
$$

Enables seamlessness!

HRAM NETS



CHAM-style operational semantics.

Denotational semantics

$$[\![S]\!] \triangleq \{s \mid \exists n.initial(S) \xrightarrow{s} n\}$$

$$\begin{aligned}
[\![S_1; S_2]\!] &= [\![S_1]\!]; [\![S_2]\!] \\
&\triangleq \{s \restriction B \mid s \in traces_{A \otimes B \otimes C} \wedge s \restriction C \in [\![S_1]\!] \wedge \pi \cdot s^{*B} \restriction A \in [\![S_2]\!]\} \\
&= \text{Synchronisation and hiding} \\
[\![S_1 \otimes S_2]\!] &= [\![S_1]\!] \otimes [\![S_2]\!] \\
&\triangleq \{s \mid s \in traces_{A \otimes B} \wedge s \restriction B \in [\![S_1]\!] \wedge s \restriction A \in [\![S_2]\!]\} \\
&= \text{Trace interleaving}
\end{aligned}$$

Theorem
HRAM nets form a symmetric compact-closed category, called ***HRAMnet****.*

Game nets

HRAM nets whose denotation corresponds to games.

Idea:
Ports $\leftrightarrow$ moves
Heap pointers $\leftrightarrow$ justification pointers

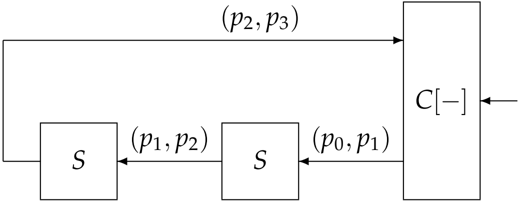
PLUG AND PLAY?

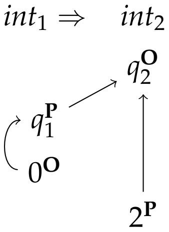
Invariant: Each (non-initial) message has one known and one unknown pointer.

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 $C[-_{int}] : int$:

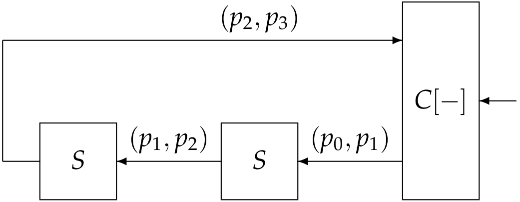


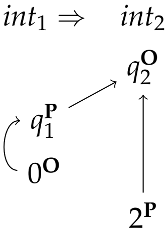


PLUG AND PLAY?

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 $C[-_{int}] : int$:



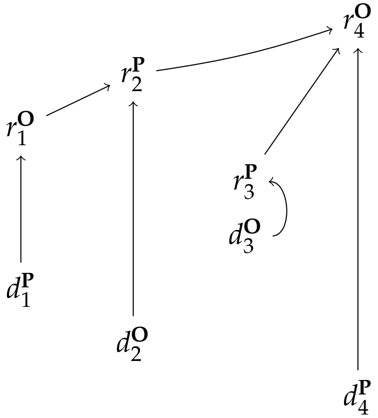


NO! The invariant breaks.

Composition must be mediated.

HRAMs for copycat

$\alpha \;:\quad (\mathbf{com}_1 \Rightarrow \mathbf{com}_2) \rightarrow (\mathbf{com}_3 \Rightarrow \mathbf{com}_4)$

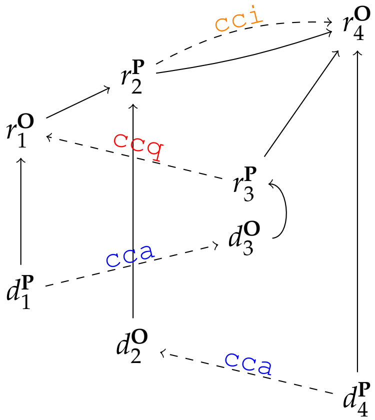


Nominally:

$$r_4 p_0[p_1] \cdot r_2 p_1[p_2] \cdot r_1 p_2[p_3] \cdot r_3 p_1[p_4] \cdot d_3 p_4 \cdots$$

HRAMs for copycat

$$\alpha \;:\quad (\mathbf{com}_1 \Rightarrow \mathbf{com}_2) \to (\mathbf{com}_3 \Rightarrow \mathbf{com}_4)$$



$$r_2 : [p_2 \mapsto p_1]$$

$$r_3 : [p_2 \mapsto p_1, p_4 \mapsto p_3]$$

$$d_1 : [p_2 \mapsto p_1]$$

$$d_4 : []$$

Nominally:

$$r_4 p_0[p_1] \cdot r_2 p_1[p_2] \cdot r_1 p_2[p_3] \cdot r_3 p_1[p_4] \cdot d_3 p_4 \cdots$$

$$\texttt{cci} \triangleq \texttt{flip}\,0,1;\ 1 \leftarrow \texttt{new}\,0,3$$

$$\texttt{ccq} \triangleq 1 \leftarrow \texttt{new}\,1,3;\ 0,3 \leftarrow \texttt{get}\,0$$

$$\texttt{cca} \triangleq \texttt{flip}\,0,1;\ 0,3 \leftarrow \texttt{get}1;\ \texttt{free}\,1$$

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

$$\begin{aligned} P \triangleq\ & \{q_2 \mapsto C;\ \texttt{spark}\,q_1 \mid q_2 \text{ initial in } A'\} \\ & \cup \{q_2 \mapsto \texttt{ccq};\ \texttt{spark}\,q_1 \mid q_2 \text{ non-initial question in } A'\} \\ & \cup \{a_2 \mapsto \texttt{cca};\ \texttt{spark}\,a_1 \mid a_2 \text{ answer in } A'\} \\ & \cup \{q_1 \mapsto \texttt{ccq};\ \texttt{spark}\,q_2 \mid q_1 \text{ question in } A\} \\ & \cup \{a_1 \mapsto \texttt{cca};\ \texttt{spark}\,a_2 \mid a_1 \text{ answer in } A\} \end{aligned}$$

GAME COMPOSITION

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

$$q_6^{\mathbf{O}}$$

$$q_4^{\mathbf{P}}$$

$$q_3^{\mathbf{O}}$$

$$q_2^{\mathbf{P}}$$

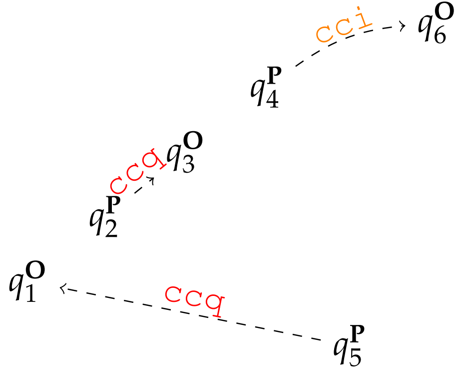
$$q_1^{\mathbf{O}}$$

$$q_5^{\mathbf{P}}$$

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

Game composition

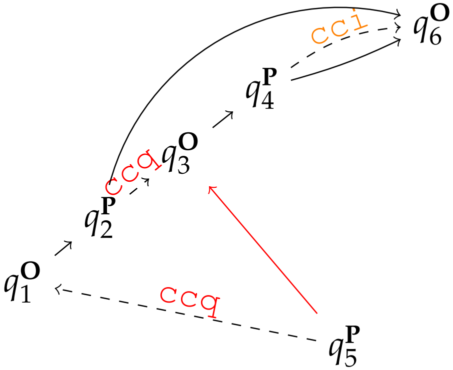
$$(A \Rightarrow B) \otimes (B' \Rightarrow C) \to (A' \Rightarrow C')$$



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

Game composition

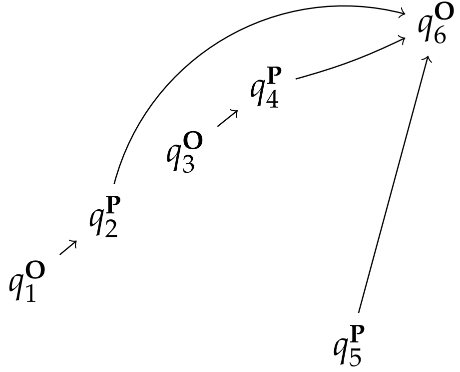
$$(A \Rightarrow B) \otimes (B' \Rightarrow C) \to (A' \Rightarrow C')$$



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

Game composition

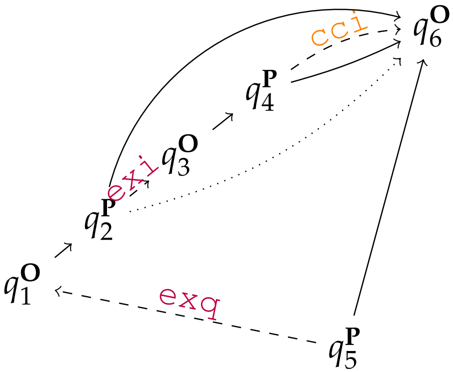
$$(A \Rightarrow B) \otimes (B' \Rightarrow C) \to (A' \Rightarrow C')$$



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

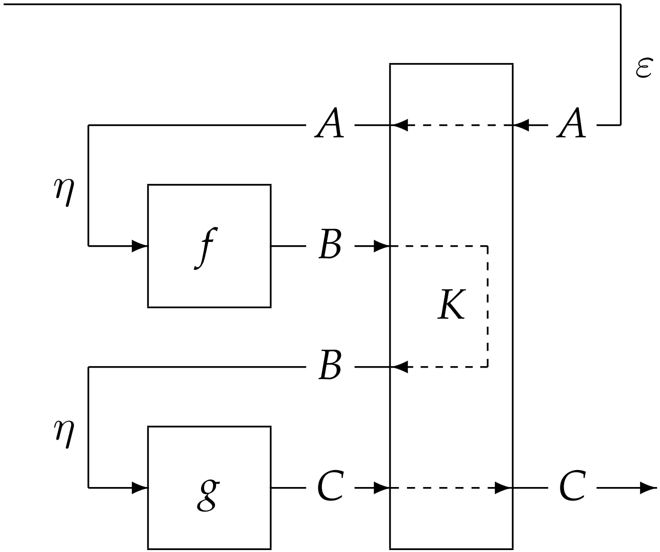
Game composition

$$(A \Rightarrow B) \otimes (B' \Rightarrow C) \to (A' \Rightarrow C')$$



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

Game composition



Game composition

Let $\mathfrak{A}'$, $\mathfrak{B}'$, and $\mathfrak{C}'$ be game interfaces such that $\pi_{\mathfrak{A}} \vdash \mathfrak{A} =_{\mathbb{A}} \mathfrak{A}'$, $\pi_{\mathfrak{B}} \vdash \mathfrak{B} =_{\mathbb{A}} \mathfrak{B}'$, $\pi_{\mathfrak{C}} \vdash \mathfrak{C} =_{\mathbb{A}} \mathfrak{C}'$, and

$$(A' \Rightarrow A, P_A) = \mathbb{C}_{\mathtt{exq},\pi_{\mathfrak{A}}^{-1},\mathfrak{A}'}$$
$$(B \Rightarrow B', P_B) = \mathbb{C}_{\mathtt{exi},\pi_{\mathfrak{B}},\mathfrak{B}}$$
$$(C \Rightarrow C', P_C) = \mathbb{C}_{\mathtt{cci},\pi_{\mathfrak{C}},\mathfrak{C}}, \text{ where}$$
$$\mathtt{exi} \triangleq 0, 3 \leftarrow \mathtt{get}\ 0;\ 1 \leftarrow \mathtt{new}\ 1, 0$$
$$\mathtt{exq} \triangleq \emptyset, 0 \leftarrow \mathtt{get}\ 0;\ 1 \leftarrow \mathtt{new}\ 1, 3$$

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

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

NEWVAR

$$
\begin{aligned}
\mathit{newvar} &\stackrel{\Delta}{=} ((\mathtt{exp}_1 \otimes (\mathtt{exp}_2 \Rightarrow \mathtt{com}_3) \Rightarrow \mathtt{exp}_4) \Rightarrow \mathtt{exp}_5, P) \\
P &\stackrel{\Delta}{=} \{q_5 \mapsto 3 \leftarrow \mathtt{set}\ 0;\ \mathtt{cci};\ \mathtt{spark}\ q_4, \\
&\qquad q_1 \mapsto \emptyset, 2 \leftarrow \mathtt{get}\ 0;\ \mathtt{flip}\ 0, 1;\ 1 \leftarrow \mathtt{set}\ \emptyset; \\
&\qquad\qquad \mathtt{spark}\ a_1, \\
&\qquad q_3 \mapsto \mathtt{flip}\ 0, 1;\ 1 \leftarrow \mathtt{new}\ 0, 1;\ \mathtt{spark}\ q_2, \\
&\qquad a_2 \mapsto \emptyset, 3 \leftarrow \mathtt{get}\ 0;\ \mathtt{update}\ 3\ 2;\ \mathtt{cca};\ \mathtt{spark}\ a_3, \\
&\qquad a_4 \mapsto \mathtt{cca};\ \mathtt{spark}\ a_5\}
\end{aligned}
$$

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 targetting 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`