AJM-games revisited

Nikos Tzevelekos

GaLoP V – Paphos – 2010
AJM vs. HO

- The Mother of All Toy Languages
- Full abstraction for PCF
- Features of two game models: HO vs. AJM
- Innocence vs. History-freedom
- A paradox
- Play-equivalence under scrutiny
- Now it all makes sense...

New-AJM games

Access Control
PCF:

- Syntax:

\[
\begin{align*}
\Gamma, x : A & \vdash x : A \\
\Gamma \vdash M : A \rightarrow B & \quad \Gamma \vdash N : A \\
\Gamma & \vdash MN : B \\
\Gamma \vdash M : \text{nat} & \quad \Gamma \vdash M : \text{nat} \\
\Gamma & \vdash M \pm N : \text{nat} \\
\Gamma \vdash M : A \rightarrow A & \\
\Gamma & \vdash Y_AM : A
\end{align*}
\]
PCF:

- Syntax:

\[
\begin{align*}
\Gamma, x : A &\vdash x : A \\
&\Gamma \vdash n : \text{nat} \\
\Gamma &\vdash M : A \rightarrow B \quad \Gamma &\vdash N : A \\
&\Gamma \vdash MN : B \\
\Gamma, x : A &\vdash M : B \\
&\Gamma \vdash \lambda x^A.M : A \rightarrow B \\
\Gamma &\vdash M : \text{nat} \quad \Gamma &\vdash M : \text{nat} \\
&\Gamma \vdash M \pm N : \text{nat} \\
\Gamma &\vdash M : \text{nat} \quad \Gamma &\vdash N, N' : A \\
&\Gamma \vdash \text{if0 } MN N' : A \\
\Gamma &\vdash M : A \rightarrow A \\
&\Gamma \vdash \text{Y} AM : A \\
(\lambda x.M)N &\rightarrow M\{N/x\} \\
\text{if0 } n N N' &\rightarrow \begin{cases} 
N & \text{if } n = 0 \\
N' & \text{if } n > 0 
\end{cases} \\
\text{Y} M &\rightarrow M(\text{Y} M) \\
\ldots
\end{align*}
\]
Full abstraction for PCF

- Solved by use of \textit{game semantics} around 1994:
  - AJM: Abramsky, Jagadeesan & Malacaria;
  - HO: Hyland & Ong and, independently, Nickau.
Solved by use of *game semantics* around 1994:

- AJM: Abramsky, Jagadeesan & Malacaria;
- HO: Hyland & Ong and, independently, Nickau.

Two different formalisms, solving the same problem.
Features of two game models: HO vs. AJM

\[ A = \langle M_A, \lambda_A, \vdash_A \rangle \] is an arena:

- \( M_A \) a set of moves.
- \( \lambda_A \) is a labelling function.
- \( \vdash_A \) is a justification relation.

\[ \lambda_A : M_A \to \{PQ, PA, OQ, OA\} \]
\[ \vdash_A \subseteq M_A \times M_A \]
Features of two game models: HO vs. AJM

\[ A = \langle M_A, \lambda_A, \vdash_A \rangle \] is an arena:

- \( M_A \) a set of moves.
- \( \lambda_A \) is a labelling function.
- \( \vdash_A \) is a justification relation.

\[ \lambda_A : M_A \rightarrow \{PQ, PA, OQ, OA\} \]
\[ \vdash_A \subseteq M_A \times M_A \]

A play is a sequence of moves with pointers (sat. Certain Conditions):

\[ q' \quad q' \quad 3 \quad q' \quad 4 \quad 7 \]
\[ O \quad P \quad O \quad P \quad O \quad P \quad P \]
\[ A = \langle M_A, \lambda_A, \vdash_A \rangle \text{ is an arena:} \]

- \( M_A \) a set of moves.
- \( \lambda_A \) is a labelling function.
- \( \vdash_A \) is a justification relation.

\[ \lambda_A : M_A \rightarrow \{PQ, PA, OQ, OA\} \]
\[ \vdash_A \subseteq M_A \times M_A \]

A play is a sequence of moves with pointers (satisfying Certain Conditions):

- **explicit pointers, non-linearity**
Features of two game models: HO vs. AJM

\[ A = \langle M_A, \lambda_A, \vdash_A \rangle \] is an \textbf{arena}:

- \( M_A \) a set of \textit{moves}.
- \( \lambda_A \) is a \textit{labelling} function.
- \( \vdash_A \) is a \textit{justification} relation.

\[ \lambda_A : M_A \to \{ P, Q, P, A, O, Q, O, A \} \]
\[ \vdash_A \subseteq M_A \times M_A \]

A \textit{play} is a sequence of moves with pointers (sat. Certain Conditions):

\[ q^* q 3 q 4 7 \]
\[ O P O P O P P \]

\textit{explicit pointers, non-linearity}

\[ A = \langle M_A, \lambda_A, P_A, \approx_A \rangle \] is a \textbf{game}:

- \( M_A \) a set of \textit{moves}.
- \( \lambda_A \) is a \textit{labelling} function.
- \( P_A \) is a set of \textit{plays}.
- \( \approx_A \) is a \textit{play-equivalence}.

\[ \lambda_A : M_A \to \{ P, Q, P, A, O, Q, O, A \} \]
\[ P_A \subseteq \{ s \in M_A^* \mid s \text{ sat. CC'} \} \]
\[ \approx_A \subseteq P_A \times P_A \]
Features of two game models: HO vs. AJM

\[ A = \langle M_A, \lambda_A, \Gamma_A \rangle \] is an arena:
- \( M_A \) a set of moves.
- \( \lambda_A \) is a labelling function.
- \( \Gamma_A \) is a justification relation.

\[ \lambda_A : M_A \rightarrow \{PQ, PA, OQ, OA\} \]
\[ \Gamma_A \subseteq M_A \times M_A \]

A play is a sequence of moves with pointers (sat. Certain Conditions):

\[ q \quad q \quad 3 \quad q \quad 4 \quad 7 \]
\[ O \quad P \quad O \quad P \quad P \quad P \]

\[ \text{explicit pointers, non-linearity} \]

\[ A = \langle M_A, \lambda_A, P_A, \approx_A \rangle \] is a game:
- \( M_A \) a set of moves.
- \( \lambda_A \) is a labelling function.
- \( P_A \) is a set of plays.
- \( \approx_A \) is a play-equivalence.

\[ \lambda_A : M_A \rightarrow \{PQ, PA, OQ, OA\} \]
\[ P_A \subseteq \{s \in M_A^* \mid s \text{ sat. } CC'\} \]
\[ \approx_A \subseteq P_A \times P_A \]

\[ q \quad q_i \quad 3_i \quad q_j \quad 4_j \quad 7 \]
\[ O \quad P \quad O \quad P \quad O \quad P \]

\[ \text{no pointers, linearity} \]
Innocence vs. History-freedom

- **Explicit pointers, non-linearity**
- **No pointers, linearity**
- The elements of a game model are *strategies*, that is, sets of plays deterministic for P.
- In particular, \( \Gamma \vdash M : A \) is a strategy on the arena/game \( \Gamma \rightarrow A \).
The elements of a game model are strategies, that is, sets of plays deterministic for $P$.

In particular, $\llbracket \Gamma \vdash M : A \rrbracket$ is a strategy on the arena/game $\llbracket \Gamma \rightarrow A \rrbracket$.

HO-strategies are innocent:

- $P$ cannot see the whole history of a play, but only a part of it, what we call the $P$-view.
The elements of a game model are strategies, that is, sets of plays deterministic for P.

In particular, \([\Gamma \vdash M : A]\) is a strategy on the arena/game \([\Gamma \rightarrow A]\).

HO-strategies are innocent:
- P cannot see the whole history of a play, but only a part of it, what we call the P-view.

AJM-strategies are history-free:
- P cannot see the whole history of a play, but only the last move.
At PCF types:

- Innocent HO-strategies precisely correspond to PCF terms.
- History-free AJM-strategies precisely correspond to PCF terms.
A paradox

At PCF types:
- Innocent HO-strategies precisely correspond to PCF terms.
- History-free AJM-strategies precisely correspond to PCF terms.
- Hence: Innocence = History-Freedom.
- But history-freedom is more general!
A paradox

At PCF types:

- Innocent HO-strategies precisely correspond to PCF terms.
- History-free AJM-stategies precisely correspond to PCF terms.

- Hence: Innocence = History-Freedom.
- But history-freedom is more general!

- Some details are missing here...
- AJM play-equivalence ($\simeq_A$) is less transparent than it seems.
At PCF types:

- Linearity of plays is overcome in arrow types by use of *indices*.
- Play-equivalence used so that choice of indices doesn’t really matter.
At PCF types:

- Linearity of plays is overcome in arrow types by use of *indices*.
- Play-equivalence used so that choice of indices doesn’t really matter.
  In fact, this should be taken with a pinch of salt...
  - A strategy $\sigma : A$ is a set of *equivalence classes of plays*.
  - The strategy’s behaviour is determined by its *skeletons*. 
At PCF types:

- Linearity of plays is overcome in arrow types by use of *indices*.
- Play-equivalence used so that choice of indices doesn’t really matter.
  
  In fact, this should be taken with a pinch of salt...

- A strategy $\sigma : A$ is a set of *equivalence classes of plays*.
- The strategy’s behaviour is determined by its *skeletons*.
- A skeleton is a set of *plays*,
- and $\sigma$ is history-free if it has a history-free skeleton.
Now it all makes sense...

- How much of a play then can we store in an index?
Now it all makes sense...

- How much of a play then can we store in an index?
- Precisely the play’s P-view!
Now it all makes sense...

- How much of a play then can we store in an index?
- Precisely the play’s P-view!

*Proof:*

- Work in AJM-games; recover pointers via *pointifixion*.
- History-free $\implies$ innocent [DHR].
- Every PCF-game is *storeful*.
- Innocent $\implies$ history-free.
New-AJM games

• New-AJM games
• Connectives pictorially
• New-AJM games

Access Control
New-AJM games

A recent re-formulation of AJM games by AJ:

- Justification is made explicit in games: $A = \langle M_A, \lambda_A, j_A, P_A, \approx_A \rangle$.
- Plays are still linear and pointer-free.
New-AJM games

A recent re-formulation of AJM games by AJ:

- Justification is made explicit in games: \( A = \langle M_A, \lambda_A, j_A, P_A, \approx_A \rangle \).
- Plays are still linear and pointer-free.
- This unifies the two trends!
A recent re-formulation of AJM games by AJ:

- Justification is made explicit in games: \( A = \langle M_A, \lambda_A, j_A, P_A, \approx_A \rangle \).
- Plays are still linear and pointer-free.
- This unifies the two trends!

Other structure:

- From AJM: \( \circ, \otimes, ! \)
- product: \& of AJM
Connectives pictorially

A

B
Connectives pictorially
Connectives pictorially

- Play at one: $A \& B$
- Play at both: $A \otimes B$
- Play at one: $A \oplus B$
- Play at both: $A \nand B$
A recent re-formulation of AJM games by AJ:

- Justification is made explicit in games: \( A = \langle M_A, \lambda_A, j_A, P_A, \approx_A \rangle \).
- Plays are still linear and pointer-free.
- This unifies the two trends!

Other structure:

- From AJM: \( \rightarrow, \otimes, ! \)
- product: \& of AJM
- par: \( \& \) of Laurent
- plus: \( \oplus \) of Laurent
- why not (\( ? \)): more technical
- negation: lifting
A recent re-formulation of AJM games by AJ:

- Justification is made explicit in games: \( A = \langle M_A, \lambda_A, j_A, P_A, \approx_A \rangle \).
- Plays are still linear and pointer-free.
- This unifies the two trends!

Other structure:

- From AJM: \( \multimap, \otimes, ! \)
- product: \( \& \) of AJM
- par: \( \otimes \) of Laurent
- plus: \( \oplus \) of Laurent
- why not (\( ? \)): more technical
- negation: lifting
- \( \ldots \) these yield a model of linear logic (i.e. linear \( \lambda\mu \)-calculus).
Access Control

AJM vs. HO

New-AJM games

Access Control
- A calculus of access control (DCC)
- Games for access control
There is a lattice \( \mathcal{L} \) of access levels.

Types (all \( l \in \mathcal{L} \)):

\[
A, B ::= \text{unit} \mid A \lor B \mid A \land B \mid A \rightarrow B \mid T_l A
\]
• There is a lattice $\mathcal{L}$ of access levels.

• Types (all $l \in \mathcal{L}$):

\[
A, B ::= \text{unit} \mid A \lor B \mid A \land B \mid A \to B \mid T_l A
\]

• Types protect access levels:

  • if $l \leq l'$ then $T_{l'} A \supseteq l$
  • if $A, B \supseteq l$ then $A \land B \supseteq l$
  • if $A \supseteq l$ then $B \to A \supseteq l$
  • unit $\supseteq l$, and if $A \supseteq l$ then $T_{l'} A \supseteq l$
A calculus of access control (DCC)

- There is a lattice $L$ of access levels.
- Types (all $l \in L$):
  \[
  A, B ::= \text{unit} \mid A \lor B \mid A \land B \mid A \rightarrow B \mid T_l A
  \]
- Types protect access levels:
  - if $l \leq l'$ then $T_{l'} A \supseteq l$
  - if $A, B \supseteq l$ then $A \land B \supseteq l$
  - if $A \supseteq l$ then $B \rightarrow A \supseteq l$
  - $\text{unit} \supseteq l$, and if $A \supseteq l$ then $T_l A \supseteq l$
- Terms: $\lambda$-calculus plus:
  \[
  \frac{\Gamma \vdash s : A}{\Gamma \vdash \eta_l s : T_l A}\quad \frac{\Gamma \vdash s : T_l A \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \text{bind } x = s \text{ in } t : B} \quad B \supseteq l
  \]
A computation of type $A$ cannot use an input of level $l$ unless $A \supseteq l$

In game semantics [AJ]:

- Moves have levels: $A = \langle M_A, \lambda_A, j_A, \text{lev}_A, P_A, \approx_A \rangle$
- E.g: $\text{lev}_{TlA}(m) = \text{lev}_A(m) \sqcup l$.
- A move can only be played if its level is below that of its justifier.
A computation of type $A$ cannot use an input of level $l$ unless $A \supseteq l$

In game semantics [AJ]:

- Moves have levels: $A = \langle M_A, \lambda_A, j_A, \text{lev}_A, P_A, \approx_A \rangle$
- E.g: $\text{lev}_{T_lA}(m) = \text{lev}_A(m) \sqcup l$.
- A move can only be played if its level is below that of its justifier.

Classical versions?

- how to allow $T_lA \rightarrow T_lA \lor B$
  and disallow $T_lA \rightarrow A \lor T_lB$