AJM-games revisited

Nikos Tzevelekos

GaLoP V – Paphos – 2010

AJM vs. HO

The Mother of All Toy Languages

PCF:

- Introduced by Plotkin (1977), embodying Scott's LCF (1969).
- Syntax:

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma \vdash n : \mathsf{nat}}$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A . M : A \to B}$$

$$\frac{\Gamma \vdash M : \mathsf{nat} \quad \Gamma \vdash M : \mathsf{nat}}{\Gamma \vdash M \pm N : \mathsf{nat}} \qquad \frac{\Gamma \vdash M : \mathsf{nat} \quad \Gamma \vdash N, N' : A}{\Gamma \vdash \mathsf{if0}\; M\; N\; N' : A}$$

$$\frac{\Gamma \vdash M : A \to A}{\Gamma \vdash \mathbf{Y}_A M : A}$$

PCF:

- Introduced by Plotkin (1977), embodying Scott's LCF (1969).
- Syntax:

$$\frac{}{\Gamma, x : A \vdash x : A}$$

$$\frac{}{\Gamma \vdash n : \mathsf{nat}}$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.M : A \to B}$$

$$\frac{\Gamma \vdash M : \mathsf{nat} \qquad \Gamma \vdash M : \mathsf{nat}}{\Gamma \vdash M \pm N : \mathsf{nat}}$$

$$\frac{\Gamma \vdash M : \mathsf{nat} \qquad \Gamma \vdash N, N' : A}{\Gamma \vdash \mathsf{if0}\, M\, N\, N' : A}$$

$$\frac{\Gamma \vdash M : A \to A}{\Gamma \vdash \mathbf{Y}_A M : A}$$

$$(\lambda x.M)N \longrightarrow M\{N/x\}$$

$$\mathsf{if0}\, n\, N\, N' \longrightarrow \begin{cases} N & \text{if } n = 0 \\ N' & \text{if } n > 0 \end{cases}$$

$$\mathbf{Y}M \longrightarrow M(\mathbf{Y}M)$$

$$\ldots$$

Full abstraction for PCF

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

Full abstraction for PCF

- Solved by use of *game semantics* around 1994:
  - AJM: Abramsky, Jagadeesan & Malacaria;
  - HO: Hyland & Ong and, indipendently, 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*.
- λ_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*.
- λ_A is a *labelling* function.
- $\vdash_A$ is a *justification* relation.

$$\lambda_A : M_A \to \{PQ, PA, OQ, OA\}$$
$$\vdash_A \subseteq M_A \times M_A$$

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

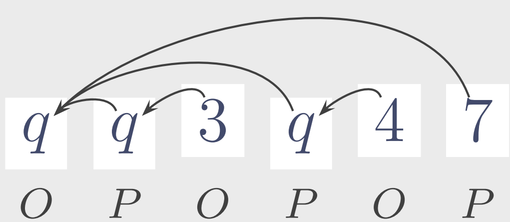
$$\begin{array}{cccccc} q & q & 3 & q & 4 & 7 \\ O & P & O & P & O & P \end{array}$$

$A = \langle M_A, \lambda_A, \vdash_A \rangle$ is an *arena*:

- M_A a set of *moves*.
- λ_A is a *labelling* function.
- $\vdash_A$ is a *justification* relation.

$$\lambda_A : M_A \to \{PQ, PA, OQ, OA\}$$
$$\vdash_A \subseteq M_A \times M_A$$

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



explicit pointers, non-linearity

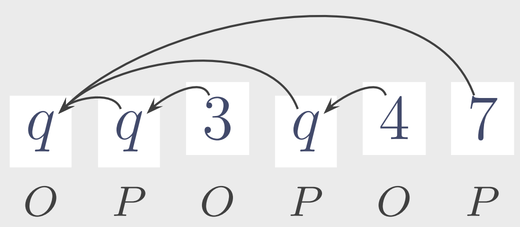
Features of two game models: HO vs. AJM

$A = \langle M_A, \lambda_A, \vdash_A \rangle$ is an *arena*:

- M_A a set of *moves*.
- λ_A is a *labelling* function.
- $\vdash_A$ is a *justification* relation.

$$\lambda_A : M_A \to \{PQ, PA, OQ, OA\}$$
$$\vdash_A \subseteq M_A \times M_A$$

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



explicit pointers, non-linearity

$A = \langle M_A, \lambda_A, P_A, \approx_A \rangle$ is a *game*:

- M_A a set of *moves*.
- λ_A is a *labelling* function.
- P_A is a set of *plays*.
- $\approx_A$ is a *play-equivalence*.

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

Features of two game models: HO vs. AJM

$A = \langle M_A, \lambda_A, \vdash_A \rangle$ is an *arena*:

- M_A a set of *moves*.
- λ_A is a *labelling* function.
- $\vdash_A$ is a *justification* relation.

$$\lambda_A : M_A \to \{PQ, PA, OQ, OA\}$$
$$\vdash_A \subseteq M_A \times M_A$$

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

$$\begin{array}{cccccc} q & q & 3 & q & 4 & 7 \\ O & P & O & P & O & P \end{array}$$

explicit pointers, non-linearity

$A = \langle M_A, \lambda_A, P_A, \approx_A \rangle$ is a *game*:

- M_A a set of *moves*.
- λ_A is a *labelling* function.
- P_A is a set of *plays*.
- $\approx_A$ is a *play-equivalence*.

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

$$\begin{array}{cccccc} q & q_i & 3_i & q_j & 4_j & 7 \\ O & P & O & P & O & P \end{array}$$

no pointers, linearity

q	q	3	q	4	7
O	P	O	P	O	P

explicit pointers, non-linearity

q	q_i	3_i	q_j	4_j	7
O	P	O	P	O	P

no pointers, linearity

Innocence vs. History-freedom

$$\begin{array}{cccccc} q & q & 3 & q & 4 & 7 \\ O & P & O & P & O & P \end{array}$$

explicit pointers, non-linearity

$$\begin{array}{cccccc} q & q_i & 3_i & q_j & 4_j & 7 \\ O & P & O & P & O & P \end{array}$$

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 $[\![\Gamma \to A]\!]$.

Innocence vs. History-freedom

$$\begin{array}{cccccc} q & q & 3 & q & 4 & 7 \\ O & P & O & P & O & P \end{array}$$

explicit pointers, non-linearity

$$\begin{array}{cccccc} q & q_i & 3_i & q_j & 4_j & 7 \\ O & P & O & P & O & P \end{array}$$

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 $[\![\Gamma \rightarrow A]\!]$.

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*.

Innocence vs. History-freedom

$$\begin{array}{cccccc} q & q & 3 & q & 4 & 7 \\ O & P & O & P & O & P \end{array}$$

explicit pointers, non-linearity

$$\begin{array}{cccccc} q & q_i & 3_i & q_j & 4_j & 7 \\ O & P & O & P & O & P \end{array}$$

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 $[\![\Gamma \to A]\!]$.

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.

A paradox

At PCF types:

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

A paradox

At PCF types:

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

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

- Some details are missing here...
- AJM play-equivalence ($\approx_A$) is less transparent than it seems.

Play-equivalence under scrutiny

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.

Play-equivalence under scrutiny

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*.

Play-equivalence under scrutiny

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 σ 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

A recent re-formulation of AJM games by AJ:

- Justification is made explicit in games: $A = \langle M_A, \lambda_A, \mathrm{j}_A, P_A, \approx_A \rangle$.
- 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: $A = \langle M_A, \lambda_A, \mathrm{j}_A, P_A, \approx_A \rangle$.
- Plays are still linear and pointer-free.
- This unifies the two trends!

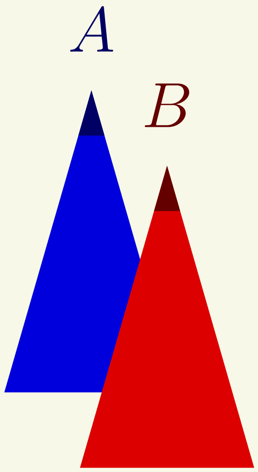
A recent re-formulation of AJM games by AJ:

- Justification is made explicit in games: $A = \langle M_A, \lambda_A, \mathrm{j}_A, P_A, \approx_A \rangle$.
- Plays are still linear and pointer-free.
- This unifies the two trends!

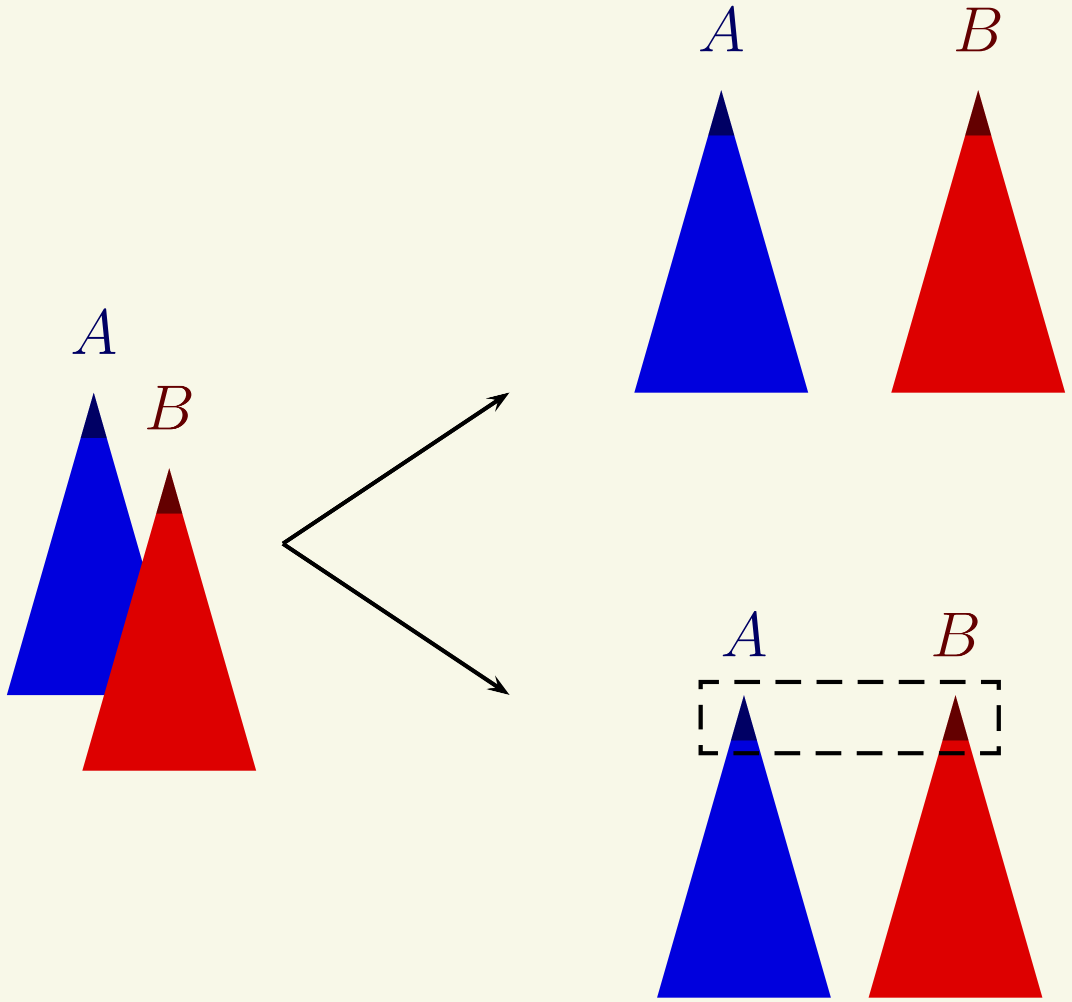
Other structure:

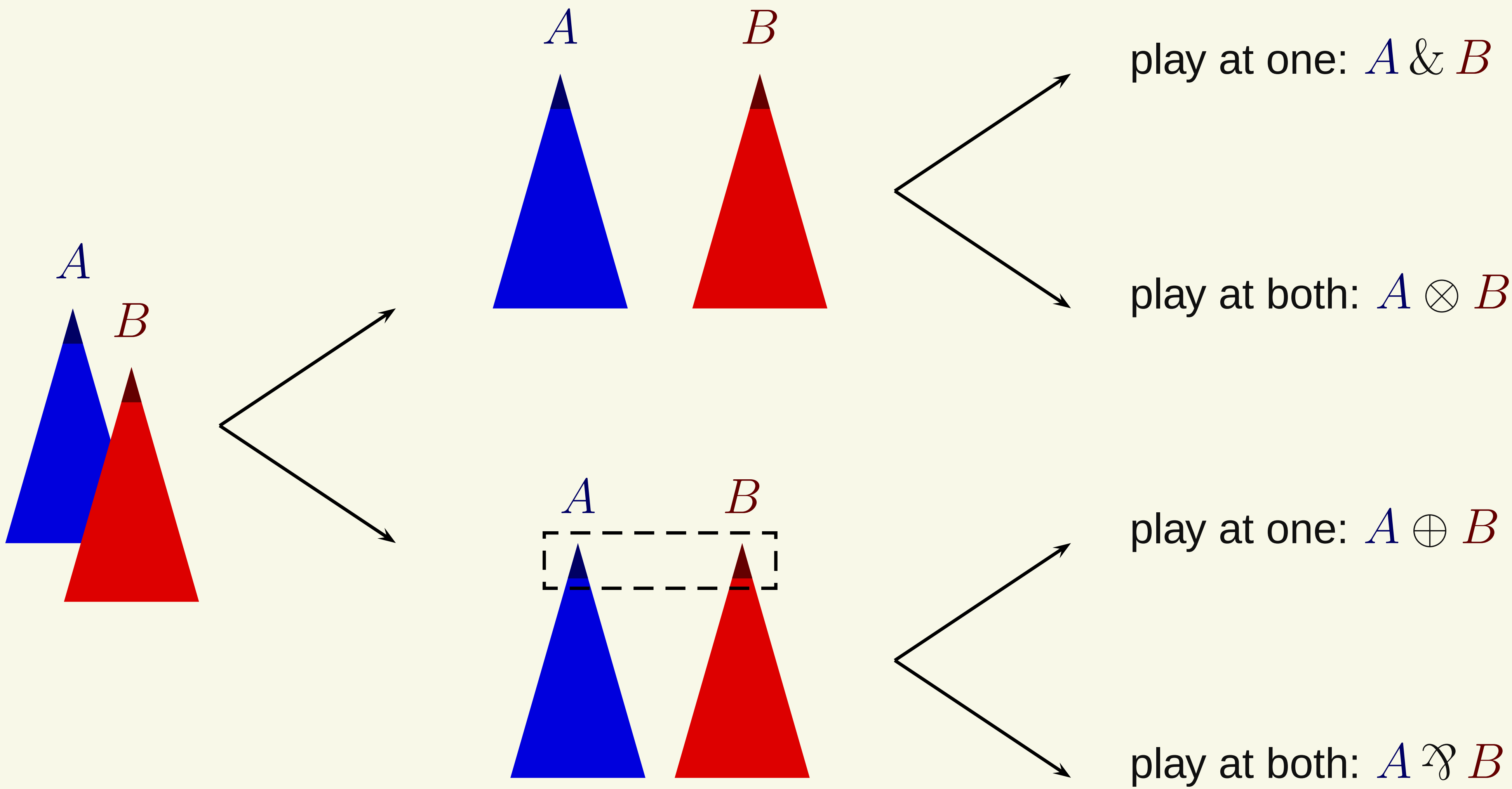
- From AJM: $\multimap$, $\otimes$, $!$
- product: $\&$ *of AJM*

Connectives pictorially



Connectives pictorially




A
B
A
B
A
B
play at one: $A \,\&\, B$
play at both: $A \otimes B$
play at one: $A \oplus B$
play at both: $A \parr B$

New-AJM games

A recent re-formulation of AJM games by AJ:

- Justification is made explicit in games: $A = \langle M_A, \lambda_A, \mathrm{j}_A, P_A, \approx_A \rangle$.
- Plays are still linear and pointer-free.
- This unifies the two trends!

Other structure:

- From AJM: $\multimap$, $\otimes$, $!$
- product: $\&$ *of AJM*
- par: $\parr$ *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: $A = \langle M_A, \lambda_A, \mathrm{j}_A, P_A, \approx_A \rangle$.
- Plays are still linear and pointer-free.
- This unifies the two trends!

Other structure:

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

Access Control

A calculus of access control (DCC)

- There is a lattice $\mathcal{L}$ of *access levels*.
- Types (all $l \in \mathcal{L}$):

$$A, B \ ::= \ \texttt{unit} \mid A \vee B \mid A \wedge B \mid A \rightarrow B \mid T_l A$$

A calculus of access control (DCC)

- There is a lattice $\mathcal{L}$ of *access levels*.
- Types (all $l \in \mathcal{L}$):

$$A, B ::= \texttt{unit} \mid A \vee B \mid A \wedge B \mid A \rightarrow B \mid T_l A$$

- Types *protect* access levels:
  - if $l \leq l'$ then $T_{l'} A \supseteq l$
  - if $A, B \supseteq l$ then $A \wedge B \supseteq l$
  - if $A \supseteq l$ then $B \rightarrow A \supseteq l$
  - $\texttt{unit} \supseteq l$, and if $A \supseteq l$ then $T_{l'} A \supseteq l$

A calculus of access control (DCC)

- There is a lattice $\mathcal{L}$ of *access levels*.
- Types (all $l \in \mathcal{L}$):

$$A, B \ ::= \ \texttt{unit} \mid A \vee B \mid A \wedge B \mid A \rightarrow B \mid T_l A$$

- Types *protect* access levels:
  - if $l \leq l'$ then $T_{l'} A \supseteq l$
  - if $A, B \supseteq l$ then $A \wedge B \supseteq l$
  - if $A \supseteq l$ then $B \rightarrow A \supseteq l$
  - $\texttt{unit} \supseteq l$, and if $A \supseteq l$ then $T_{l'} A \supseteq l$
- Terms: λ-calculus plus:

$$\frac{\Gamma \vdash s : A}{\Gamma \vdash \eta_l s : T_l A} \qquad \frac{\Gamma \vdash s : T_l A \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \texttt{bind}\ x = s\ \texttt{in}\ t : B}\ B \supseteq l$$

A computation of type A cannot use an input of level l unless $A \supseteq l$

In game semantics [AJ]:

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

Games for access control

$\boxed{\textit{A computation of type } A \textit{ cannot use an input of level } l \textit{ unless } A \supseteq l}$

In game semantics [AJ]:

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

Classical versions?

- how to allow $T_l A \longrightarrow T_l A \ \vee \ B$

  and disallow $T_l A \longrightarrow A \ \vee \ T_l B$