Towards Nominal Abramsky

Andrzej Murawski

University of Warwick

Nikos Tzevelekos

Queen Mary, U. of London

What this talk is about

Abramsky's cube (1990's):
a taxonomy of game semantics models

Nominal game semantics (2000's):
games for programs generating new/fresh resources (references, exceptions, channels, etc.)

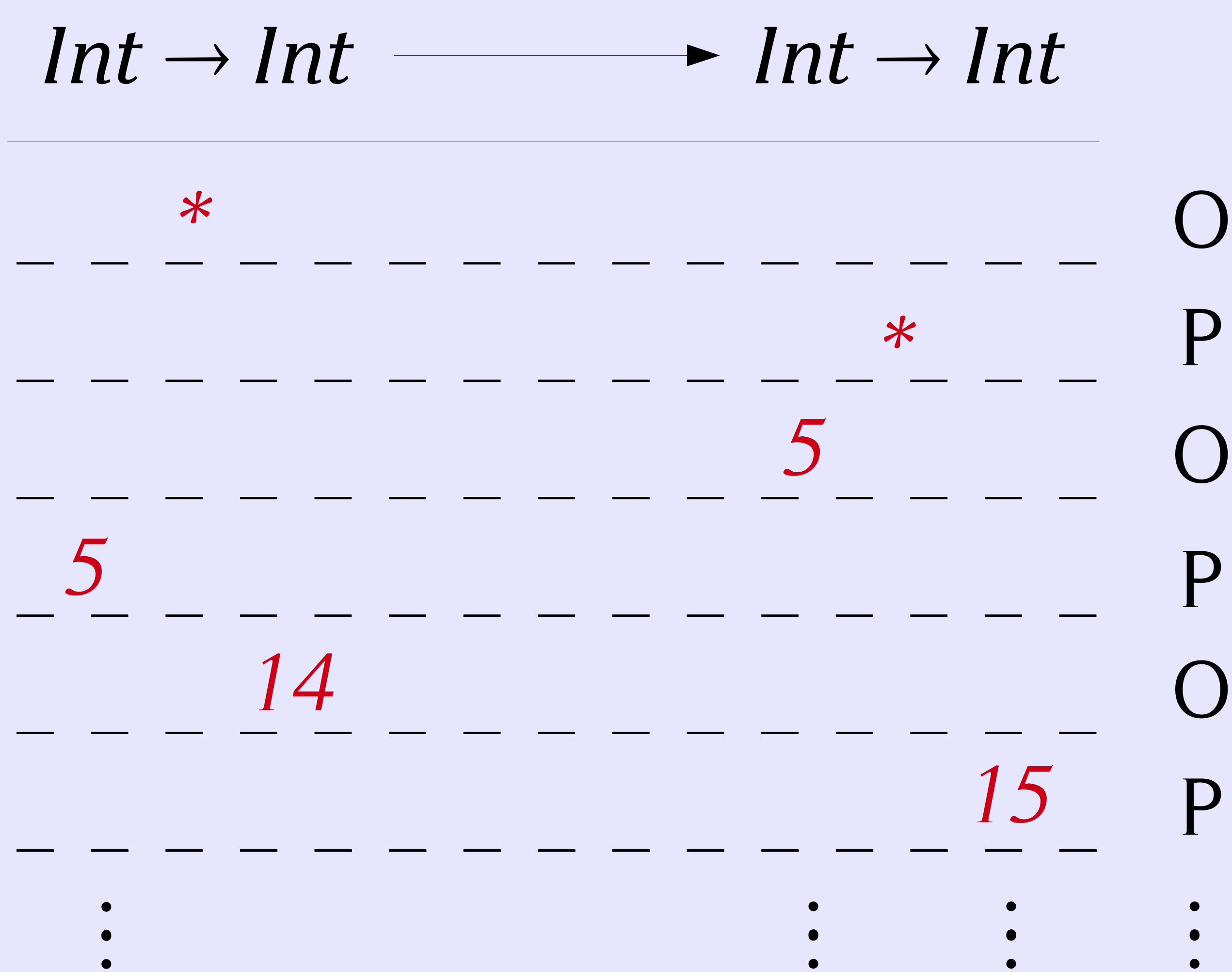
Nominal Abramsky:
the construction of an analogous taxonomy for nominal game models

Game Semantics

- Computation is modelled as a 2-player game between:
  - *Opponent* (the environment)
  - *Proponent* (the program)
- Qualitative games ($\neq$ Game Theory)
- Programs = *strategies* for Proponent
- Categories of games

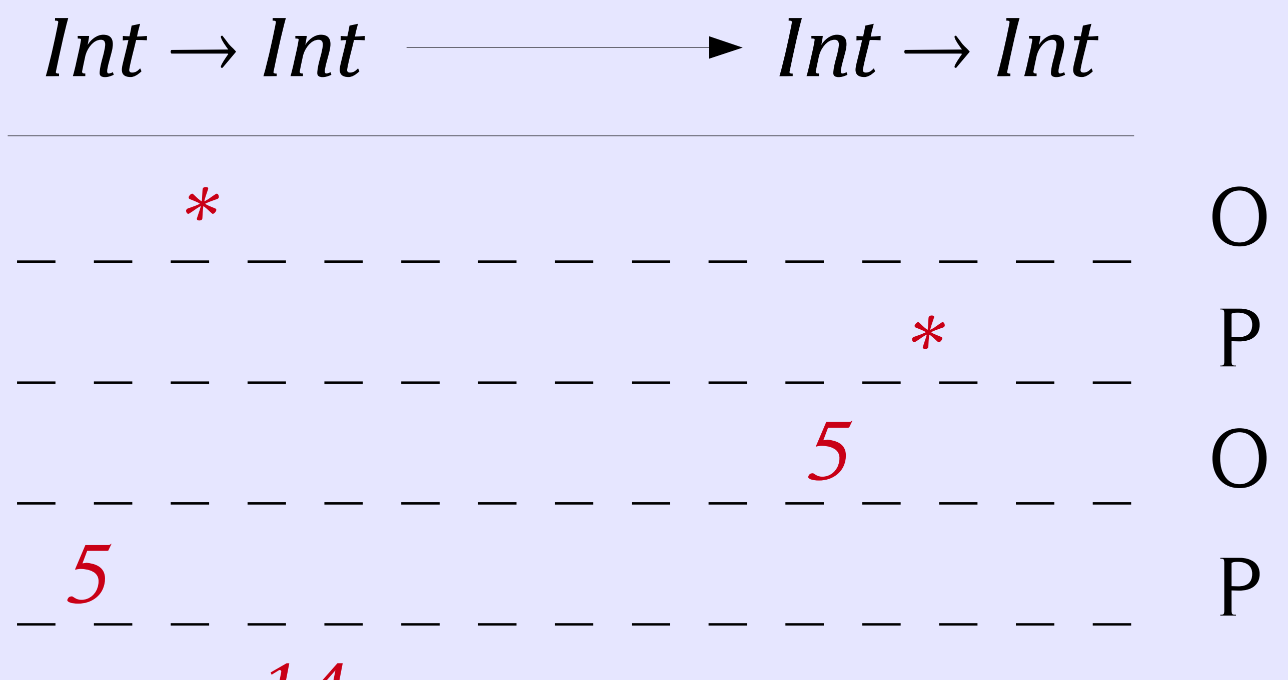
Example strategy

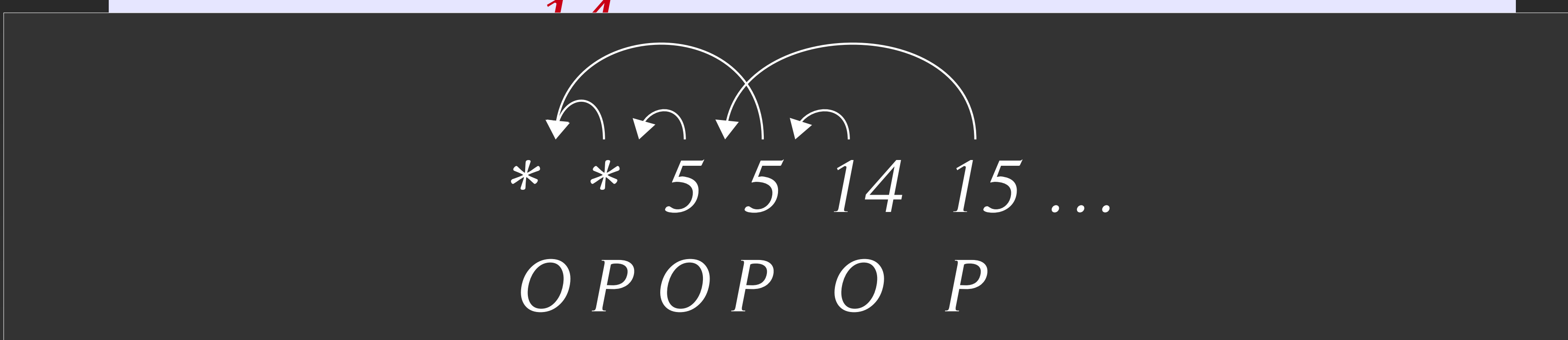
$\texttt{f : int}\to\texttt{int} \vdash \lambda\texttt{x.f(x)+1 : int}\to\texttt{int}$



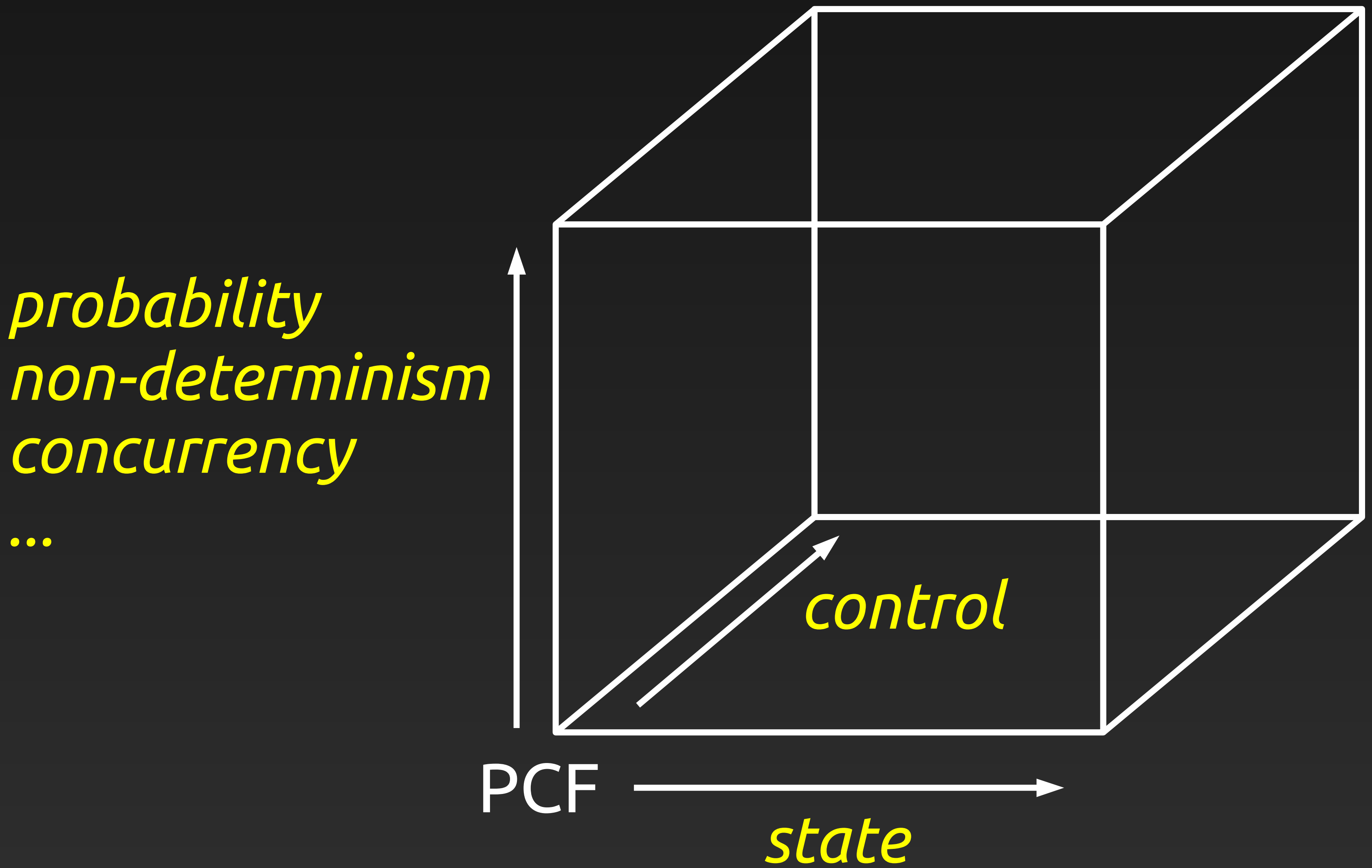
Example strategy

$f : \texttt{int} \to \texttt{int} \vdash \lambda x.f(x)+1 : \texttt{int} \to \texttt{int}$





Abramsky's cube (90's)



move at each axis: relax constraints

Two ways to model references

Reynolds

- *Idealized Algol (1978)*

References are *pairs:*

```
ref int =
(unit → int)x(int → unit)
```

⟼ $(\mathbf{1} \to \mathbf{Z}) \times (\mathbf{Z} \to \mathbf{1})$

- Theoretically attractive
- but: `mkvar(λx.3,λx.())` (*bad variables*)

Two ways to model references

Reynolds
- *Idealized Algol (1978)*

References are *pairs:*

`ref int =`
`(unit → int)x(int → unit)`

$\longmapsto$ **(1 → Z)x(Z → 1)**

- Theoretically attractive
- but: `mkvar(λx.3,λx.())` (*bad variables*)

Pitts & Stark
- *nu-calculus (1993)*

References are *names:*

`ref int = base type`

$\longmapsto$ N (names)

- Notion of *resource (name):*
  - atomic values
  - infinitely many
  - comparable for equality

Two ways to model references exceptions channels …

Reynolds

- *Idealized Algol (1978)*

References are *pairs:*

```
ref int =
(unit → int)x(int → unit)
```

$\longmapsto$ $(\mathbf{1} \to \mathbf{Z}) \times (\mathbf{Z} \to \mathbf{1})$

- Theoretically attractive
- but: `mkvar(λx.3,λx.())` (*bad variables*)

Pitts

- *nu-* …)

References are *names:*

```
ref int = base type
```

$\longmapsto$ N (names)

- Notion of *resource (name):*
  - atomic values
  - infinitely many
  - comparable for equality

Nominal games

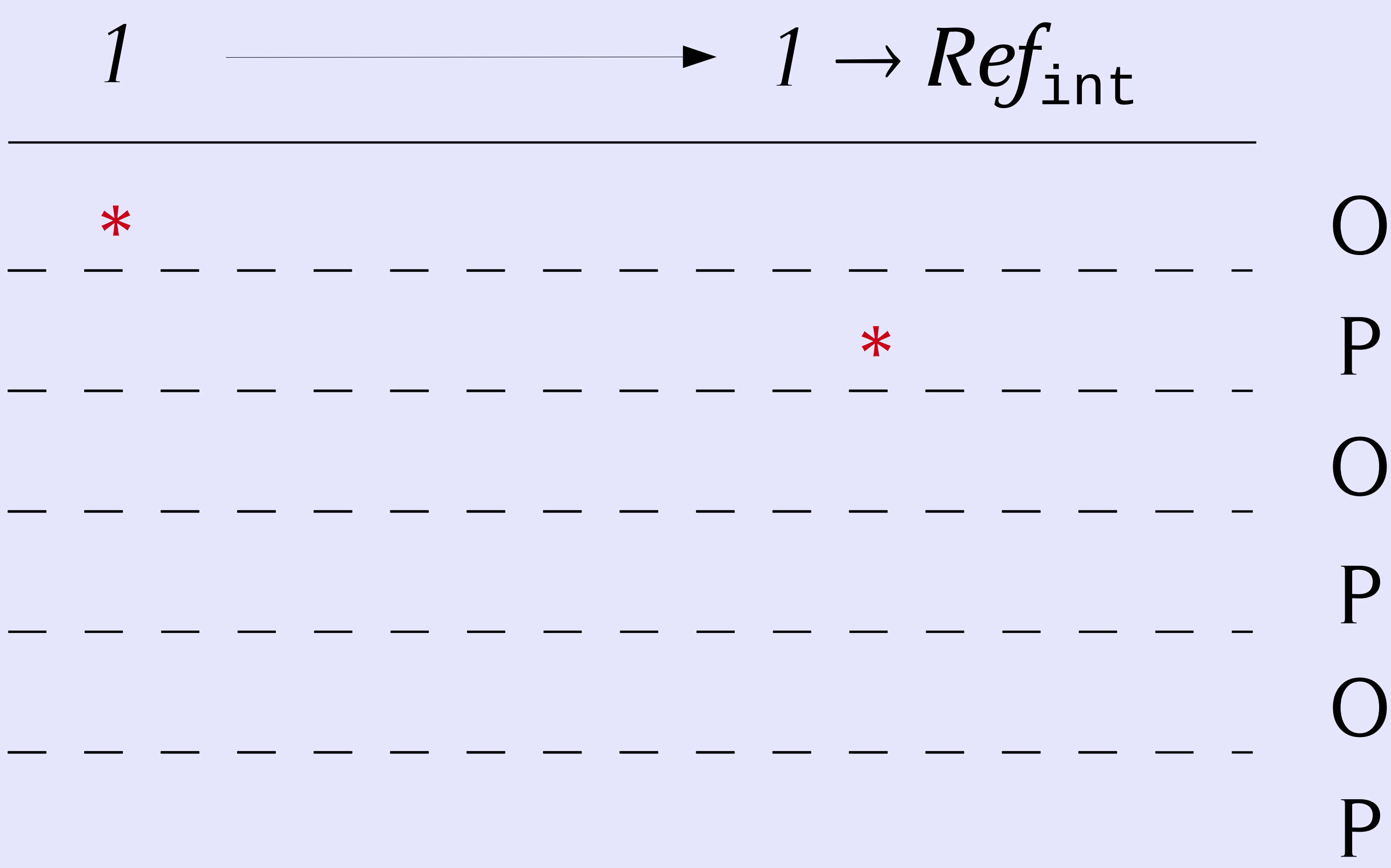
`λx.ref(0)` : `unit` → `ref int`

$$1 \longrightarrow 1 \to \mathit{Ref}_{\texttt{int}}$$

1		$1 \to \mathit{Ref}_{\texttt{int}}$	
*			O
			P
			O
			P
			O
			P

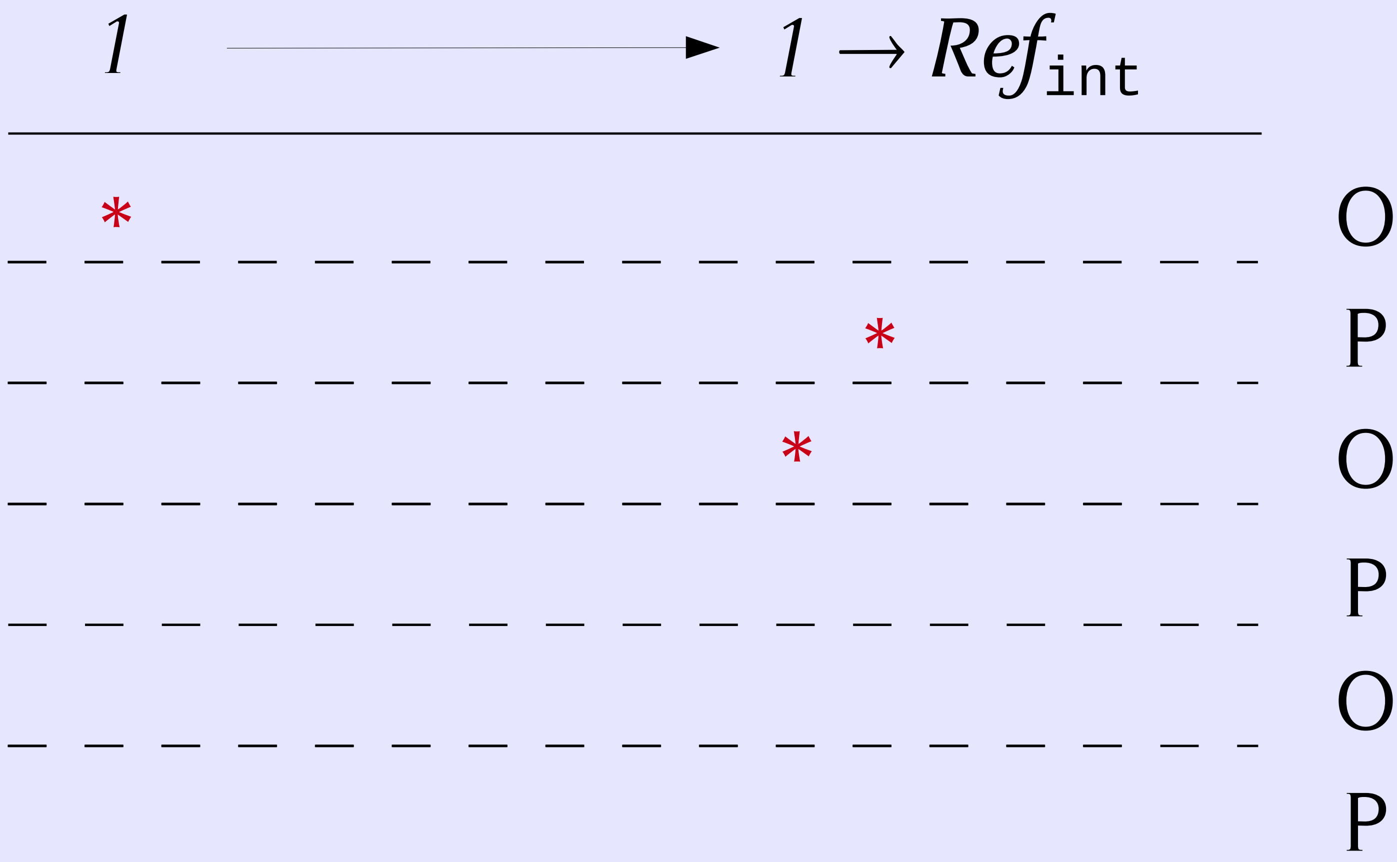
Nominal games

`λx.ref(0) : unit → ref int`



Nominal games

`λx.ref(0) : unit → ref int`



Nominal games

`λx.ref(0)` : `unit` → `ref int`

1	$\longrightarrow$	$1 \to Ref_{\mathtt{int}}$	
$*$			O
		$*$	P
		$*$	O
		$a_1^{(a_1,0)}$	P
			O
			P

Nominal games

`λx.ref(0) : unit → ref int`

1	$1 \rightarrow Ref_{\texttt{int}}$	
$*$		O
	$*$	P
	$*$	O
	$a_1^{(a_1,0)}$	P
	$*$	O
		P

Nominal games

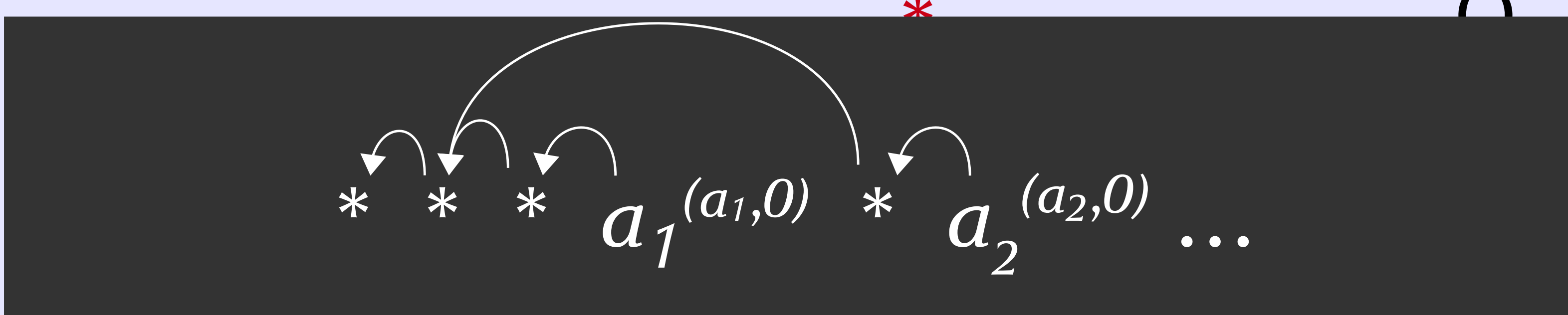
```
λx.ref(0) : unit → ref int
```

1	1	$\rightarrow$	$Ref_{\mathtt{int}}$	
$*$				O
		$*$		P
	$*$			O
			$a_1^{(a_1,0)}$	P
	$*$			O
			$a_2^{(a_2,0)}$	P
	$\vdots$		$\vdots$	$\vdots$

Nominal games

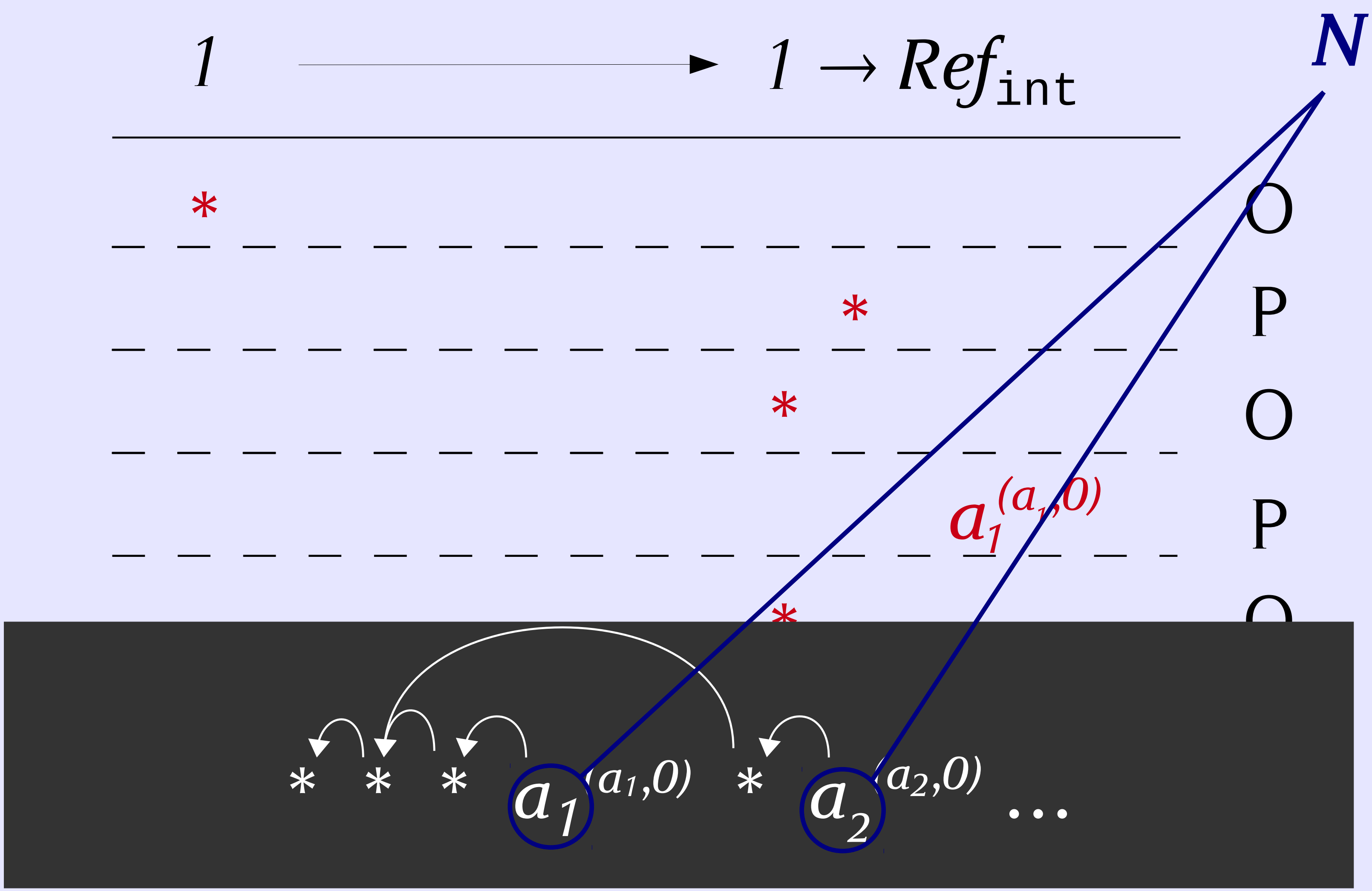
`λx.ref(0) : unit → ref int`

1	$1 \to Ref_{\texttt{int}}$	
$*$		O
	$*$	P
	$*$	O
	$a_1^{(a_1,0)}$	P
	$*$	O

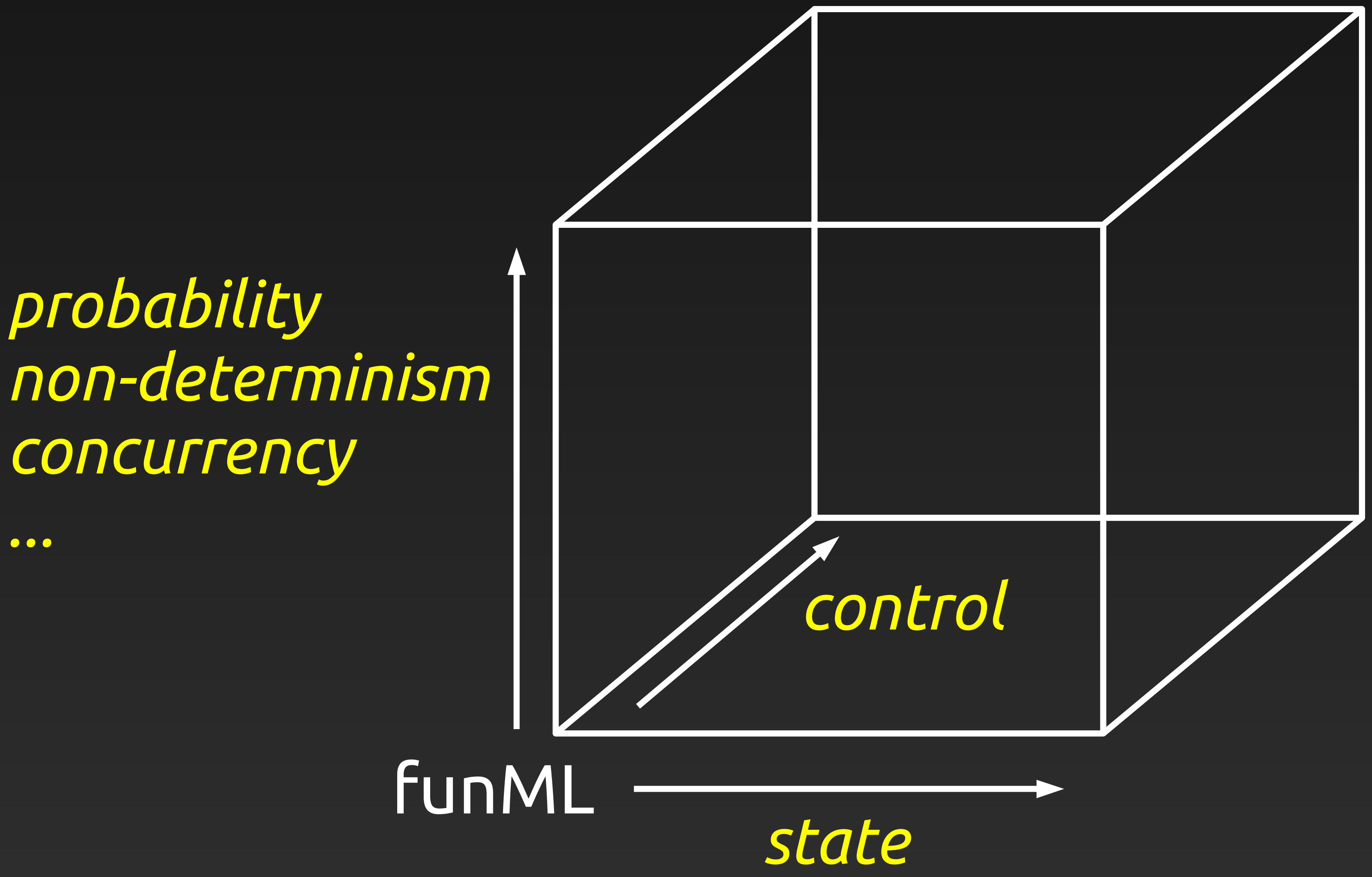


Nominal games

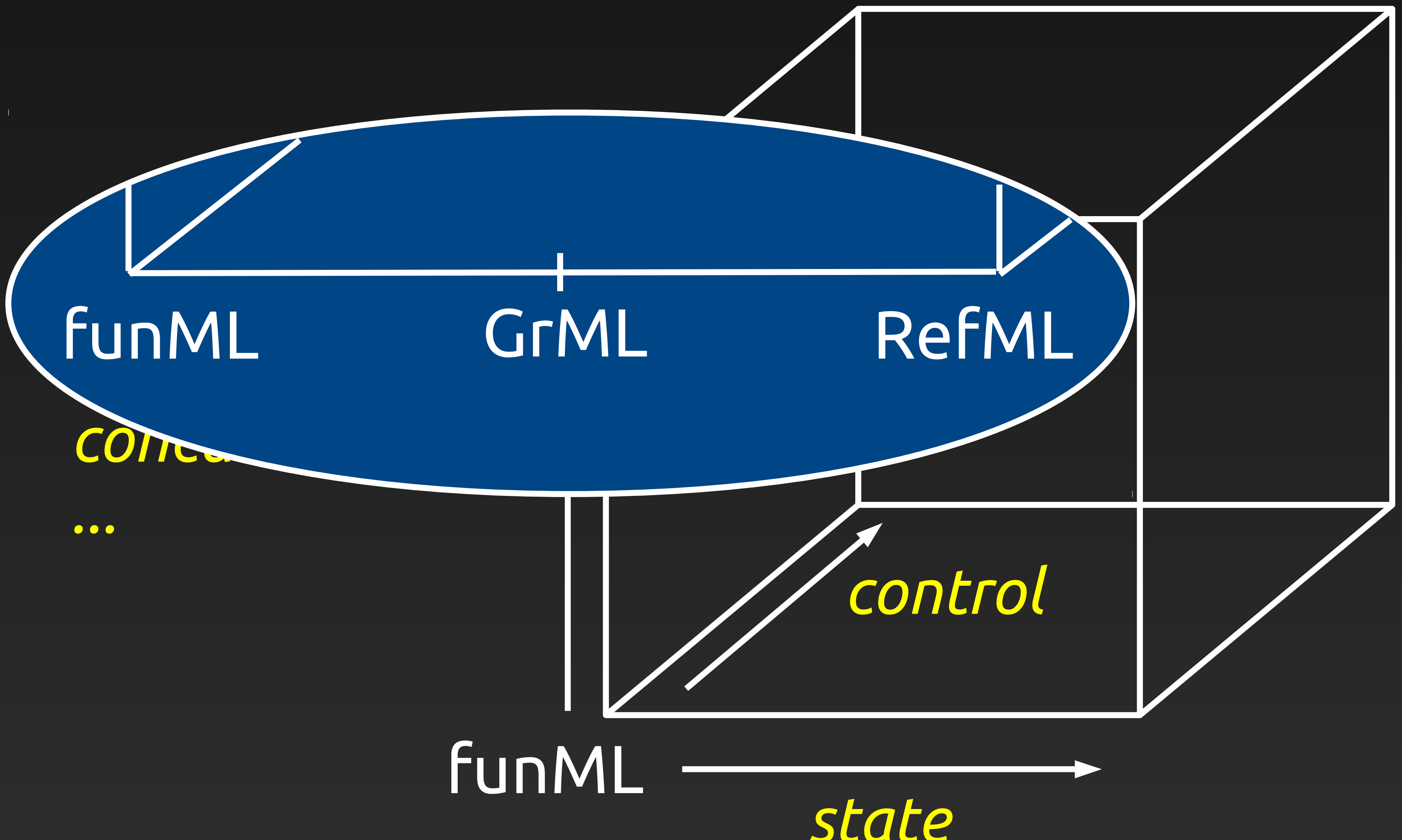
λx.ref(0) : unit → ref int



Towards nominal Abramsky



Towards nominal Abramsky



RefML: storable functions

$$A ::= \texttt{unit} \mid \texttt{int} \mid \texttt{ref}\, A \mid A \to A$$

$$\frac{\Gamma, x{:}A \vdash s : B}{\Gamma \vdash \lambda x.s : A \to B} \qquad \frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A}{\Gamma \vdash s\,t : B}$$

$$\frac{}{\Gamma \vdash ()\!:\!\texttt{unit},\ i\!:\!\texttt{int}} \qquad \frac{\Gamma \vdash s : A}{\Gamma \vdash \texttt{ref}(s) : \texttt{ref}\, A}$$

$$\frac{\Gamma \vdash s : \texttt{ref}\, A}{\Gamma \vdash\, !s : A} \qquad \frac{\Gamma \vdash s{:}\texttt{ref}\, A,\ t{:}A}{\Gamma \vdash s{:=}t : \texttt{unit}} \qquad \frac{\Gamma \vdash s, t{:}\texttt{ref}\ A}{\Gamma \vdash s == t : \texttt{int}}$$

Game semantics for RefML

MTz LICS'11

- Moves with HO-store

$$S = \{\ (a,4),\ (b,c),\ (c,3),\ (d,*),\ \ldots\ \}$$

- Justify by store

$$m^{(a,*)} \ldots m'$$

- Frugality

$$\ldots\ m^{\{(a,v),\ \ldots\}} \quad \Rightarrow \quad \ldots\ a^{S} \ldots\ m^{\{(a,v),\ \ldots\}}$$

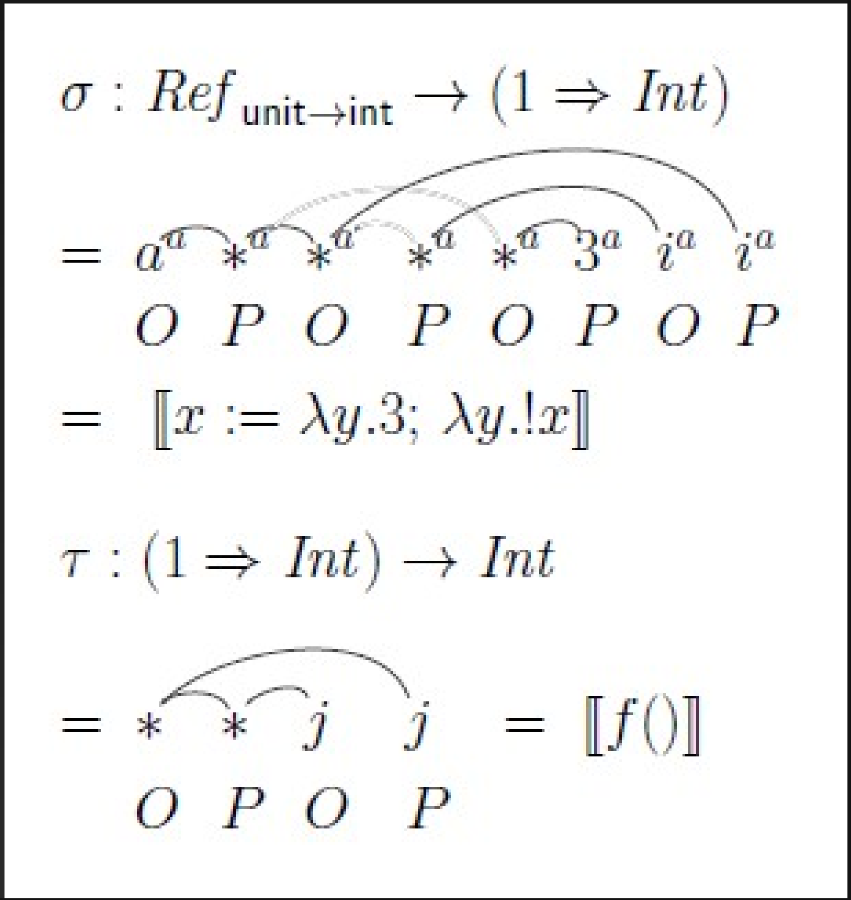
Game semantics for RefML

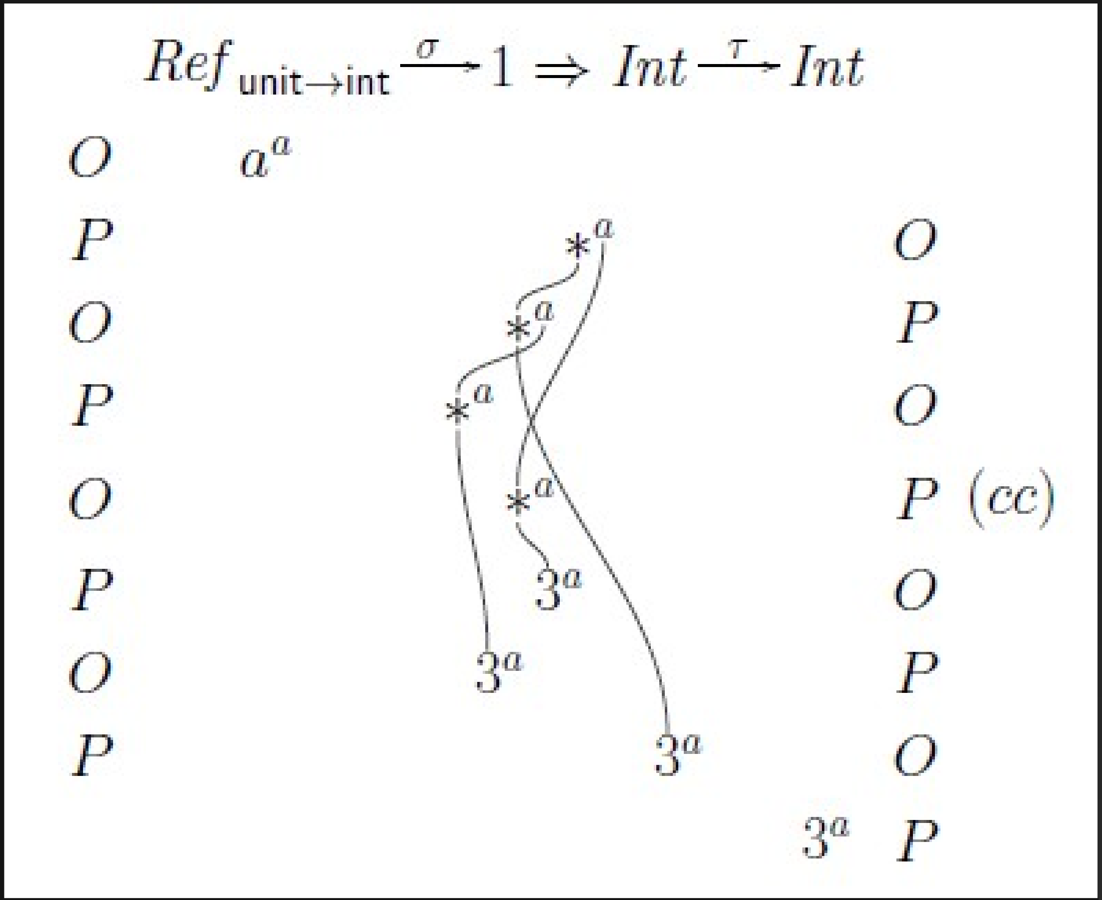
$$\sigma : Ref_{\mathsf{unit} \to \mathsf{int}} \to (1 \Rightarrow Int) \;=\; \begin{matrix} a^a & *^a & *^a & *^a & *^a & 3^a & i^a & i^a \\ O & P & O & P & O & P & O & P \end{matrix}$$

$$= [\![x : \mathsf{ref}\,(\mathsf{unit} \to \mathsf{int}) \vdash x := \lambda y.3;\ \lambda y.(!x)()]\!]$$

$$\tau : (1 \Rightarrow Int) \to Int \;=\; \begin{matrix} * & * & j & j \\ O & P & O & P \end{matrix} \;=\; [\![f : \mathsf{unit} \to \mathsf{int} \vdash f() : \mathsf{int}]\!]$$

Game semantics for RefML





Game semantics for RefML

$$\sigma : Ref_{\mathsf{unit}\to\mathsf{int}} \to (1 \Rightarrow Int)$$

$$= \; a^a \; *^a \; *^a \; *^a \; *^a \; 3^a \; i^a \; i^a$$

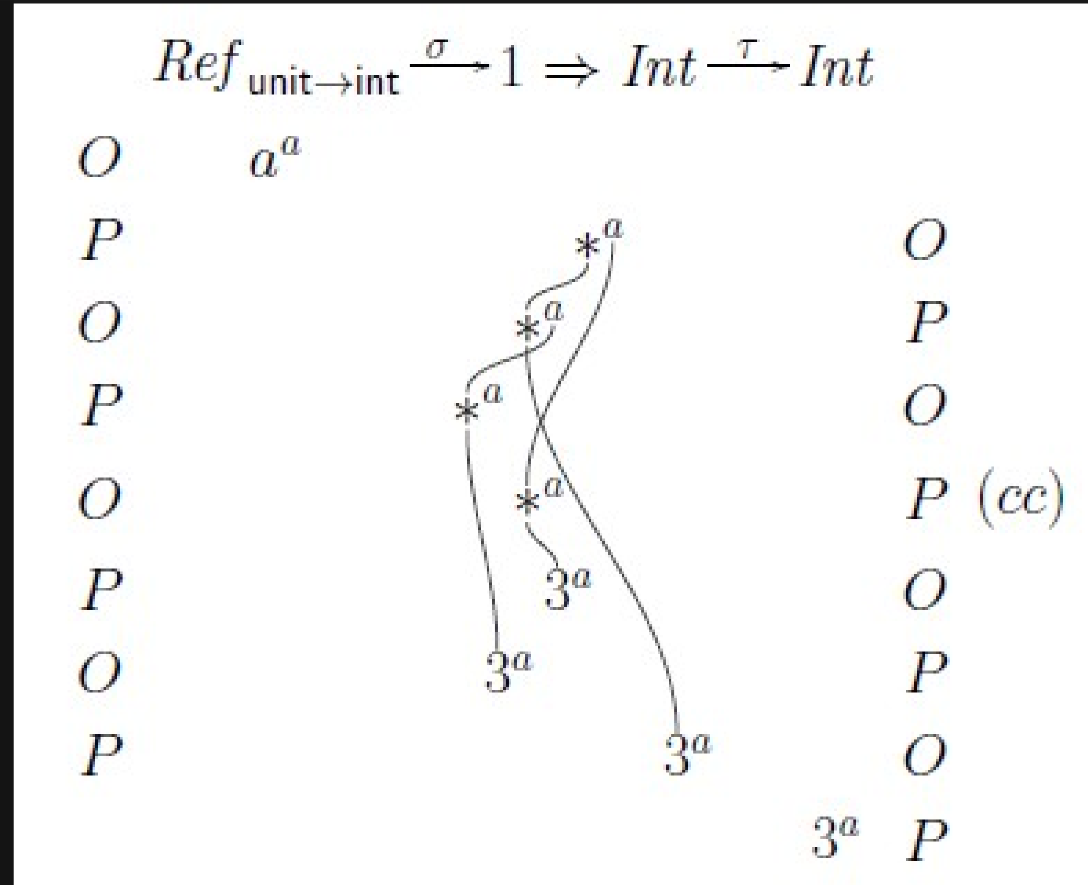
$$\quad O \; P \; O \; P \; O \; P \; O \; P$$

$$= \; [\![x := \lambda y.3;\ \lambda y.!x]\!]$$

$$\tau : (1 \Rightarrow Int) \to Int$$

$$= \; * \; * \; j \; j \quad = \; [\![f()]\!]$$

$$\quad O \; P \; O \; P$$



let $f = (x := \lambda y.3;\ \lambda y.!x)$ in $f() \;\cong\; x := \lambda y.3;\ 3$

Ground ML: full ground store

Restrict `ref` *constructor to non-function types*

- `y:ref(int→int)` $\vdash S$ but not `(λy.`S`)(ref(λx.x))`
- allow `(λy.`S`)(ref(ref(0)))`, ...

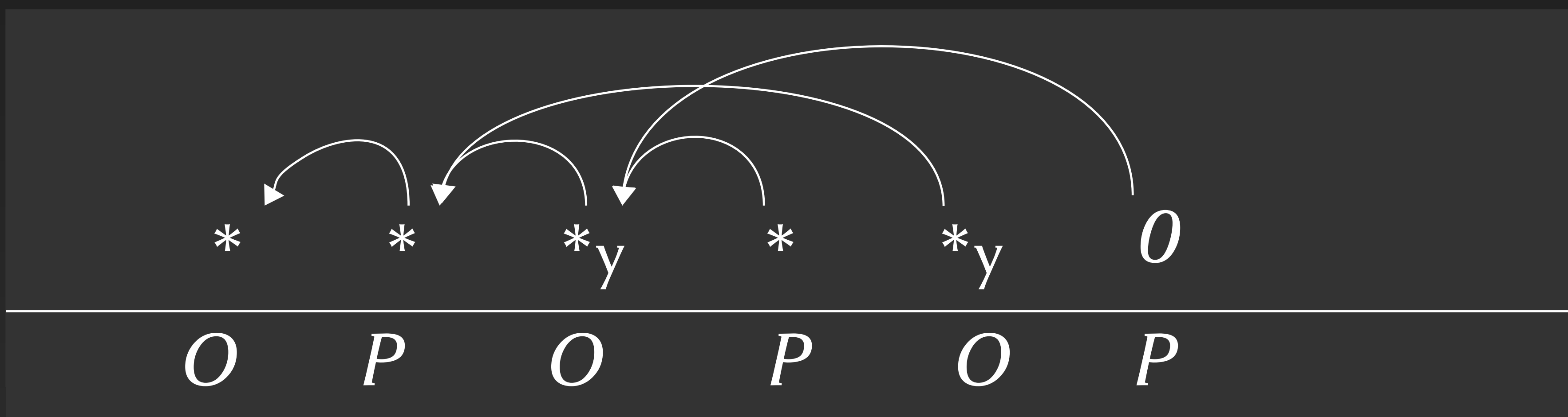
In the game model:

- Ban P from introducing/creating $(a,*)$
- Impose visibility

Visibility (breaking of)

let x=ref(..) in λy$^{\text{int}\to\text{unit}}$. first: x:=y

after: (!x) 0



Fun ML: pure functional behaviour

Remove `ref` *constructor*

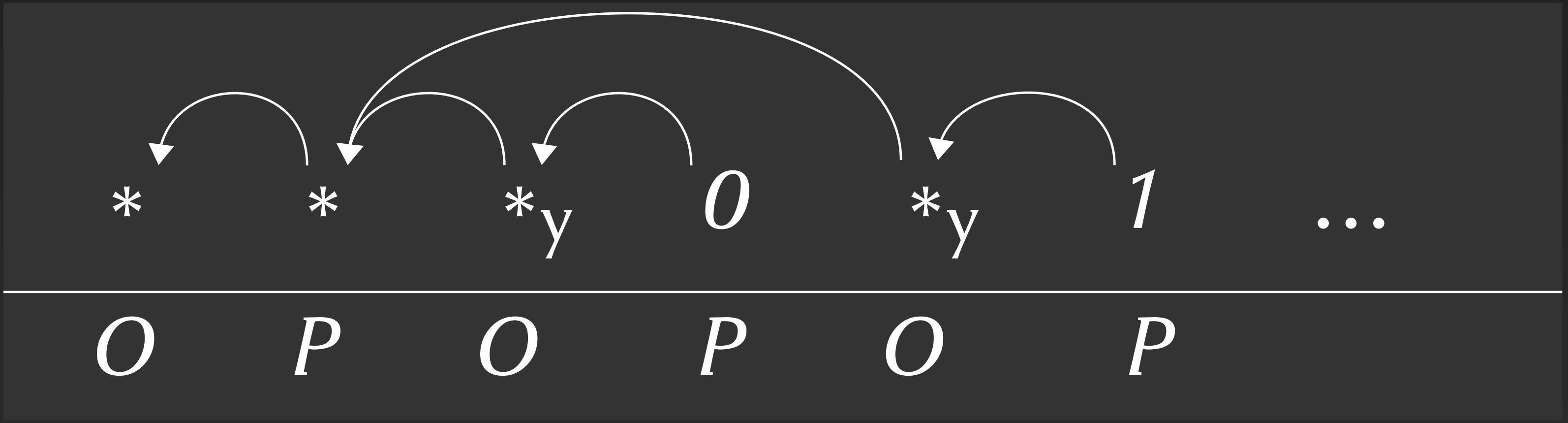
- `y:ref(int)` $\vdash S$ but not `(λy.`S`)(ref(0))`

In the game model:

- Ban P from introducing/creating any name
- Impose innocence

Innocence (breaking of)

`let x=ref(-1) in λy. x++; !x : unit → int`



Factorisations

- RefML = GrML
  + one reference of type `unit` $\rightarrow$ `unit`
  + name generators for HO-types

- GrML = FunML
  + one reference of type `int`
  + name generators for base types
  + oracles mapping names to integers

Further axes

- Concurrent ML: Laird (FSTTCS'06)

To do:

- Exceptions: Laird (LICS'01), Tz (PhD'09)
- Non-determinism: Harmer & McCusker (LICS'99)
- Polymorphism: Hughes (LICS'97), Abramsky & Jagadeesan (FOSSACS'03), Laird (LICS'10, ICALP'10)
- Probability: Danos & Harmer (LICS'00)