Nominal games: a semantics paradigm for effectful languages

Nikos Tzevelekos, QMUL

jointly with:

Andrzej Murawski, Warwick

Steven Ramsay, Oxford

Dan Ghica, Birmingham

Guilhem Jaber, Paris Diderot

Thomas Cuvillier, QMUL

PLAS seminar, U. of Kent, November 2016

what this talk is about

Examine higher-order programming languages with effects: state, exceptions, polymorphism

Look into semantic models capturing these effects

$$[\![\_]\!] : \text{Syntax} \to \mathcal{M}$$

We present nominal game semantics, a technique whereby:

- programs are modelled via 2-player games
- games use names to capture effects
- modelling is sound and fully abstract

Setting: HO programs + effects

Simply-typed λ-calculus

+ effects (HO-state, exceptions, polymorphism)

Setting: HO programs + effects

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \qquad \frac{}{\Gamma \vdash i : \mathsf{int}} \qquad \frac{}{\Gamma \vdash \mathsf{if}_{\vartheta} : \mathsf{int} \to \vartheta \to \vartheta \to \vartheta}$$

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

$$\vartheta, \vartheta' ::= \mathsf{unit} \mid \mathsf{int} \mid \vartheta \to \vartheta'$$

Setting: HO programs + effects

typed locations
(*location names*)

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \qquad \frac{}{\Gamma \vdash i : \mathsf{int}} \qquad \frac{}{\Gamma \vdash \mathsf{if}_\vartheta : \mathsf{int} \to \vartheta \to \vartheta \to \vartheta} \qquad \frac{a \in \mathsf{Loc}_\vartheta}{\Gamma \vdash a : \mathsf{ref}\ \vartheta}$$

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

$$\frac{\Gamma \vdash M, N : \mathsf{ref}\ \vartheta}{\Gamma \vdash M = N : \mathsf{int}}$$

$$\vartheta, \vartheta' ::= \mathsf{unit} \mid \mathsf{int} \mid \vartheta \to \vartheta' \mid \mathsf{ref}\ \vartheta$$

$$\frac{\Gamma \vdash M : \vartheta}{\Gamma \vdash \mathsf{ref}\ M : \mathsf{ref}\ \vartheta} \qquad \frac{\Gamma \vdash M : \mathsf{ref}\ \vartheta}{\Gamma \vdash !M : \vartheta} \qquad \frac{\Gamma \vdash M : \mathsf{ref}\ \vartheta \quad \Gamma \vdash N : \vartheta}{\Gamma \vdash M := N : \mathsf{unit}}$$

$$v ::= () \mid i \mid \mathsf{if} \mid a \mid \lambda x . M$$

Operational semantics (examples)

$$M,\ S \quad \rightarrow \quad M',\ S'$$

S stores locations + their values

$$(\lambda x.M)v,\ S \ \rightarrow \ M[v/x],\ S$$

$$a = a',\ S \ \rightarrow \ 0/1,\ S$$

$a,a' \in \mathsf{Loc}_\theta$

$$\mathsf{ref}\ 0,\ S \ \rightarrow \ a,\ S \uplus (a \mapsto 0)$$

$a \in \mathsf{Loc}_{\mathsf{int}}$

$$\mathsf{ref}\ (\lambda x.x{+}1),\ S \ \rightarrow \ a,\ S \uplus (a \mapsto \lambda x.x{+}1)$$

$a \in \mathsf{Loc}_{\mathsf{int}\to\mathsf{int}}$

$\mathsf{let}\ f = \mathsf{ref}\ (\lambda x.x)\ \mathsf{in}\ f := \lambda x.(!f)x;\ (!f)5$

$$\rightarrow \ldots \rightarrow \ (!a)5,\ (a \mapsto \lambda x.(!a)x) \rightarrow \ (\lambda x.(!a)x)5,\ (a \mapsto \lambda x.(!a)x) \rightarrow \ldots$$

Modelling problem

Build model $\mathcal{M}$ and denotation map

$$[\![\_]\!] : \text{Syntax} \longrightarrow \mathcal{M}$$

such that the modelling is sound and complete:

$$[\![P]\!] = [\![P']\!] \iff P \cong P'$$

Why?

- reasoning about programs → model checking
  [e.g. model checking equivalence, see Coneqct]

$P \cong P'$: *same observable behaviour in every context*

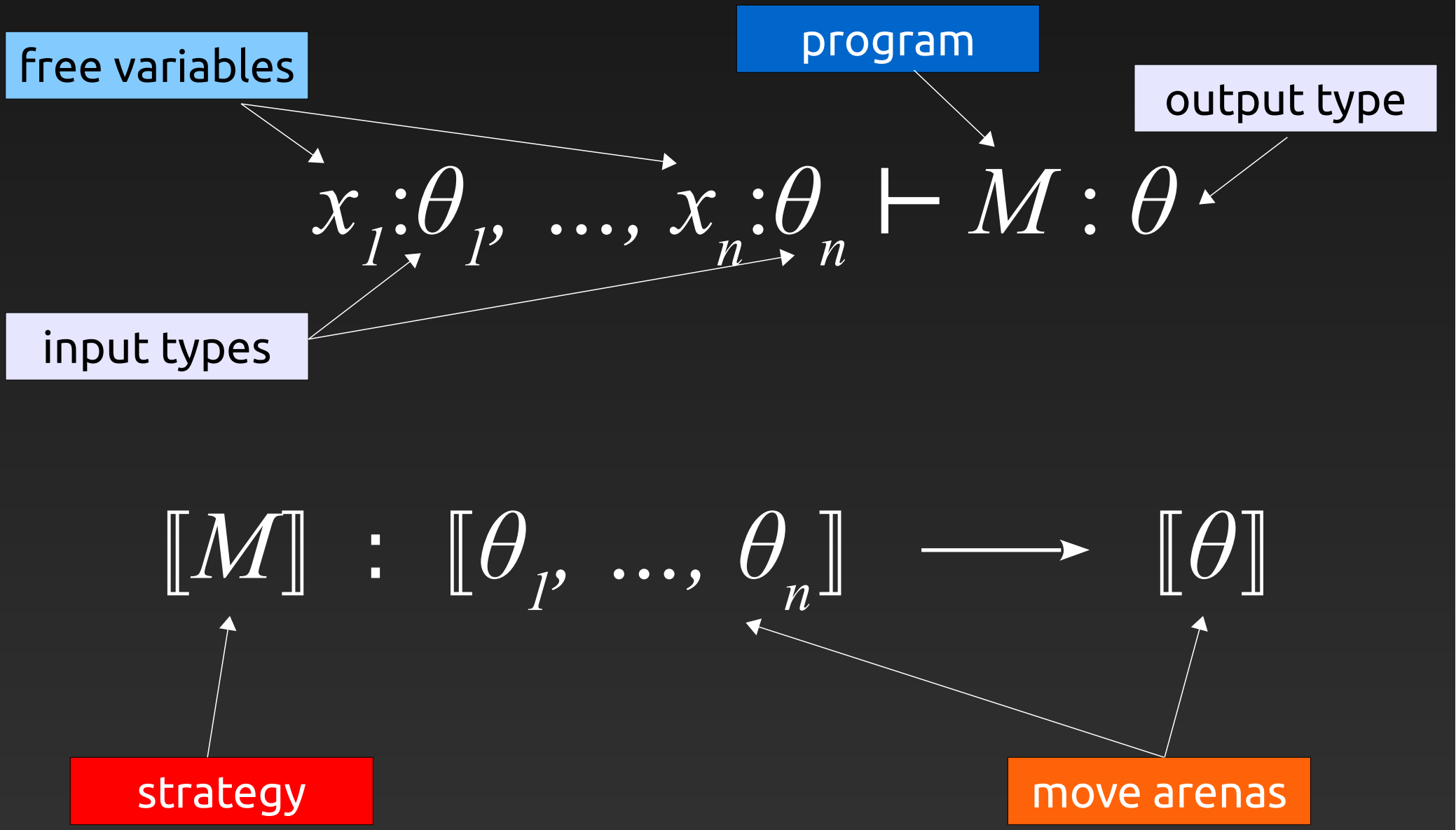
Modelling using games

- Computation is a 2-player game between:
  - *Opponent* (the environment), aka O
  - *Proponent* (the program), aka P
- Programs = *strategies* for P
- Categories of games via composition

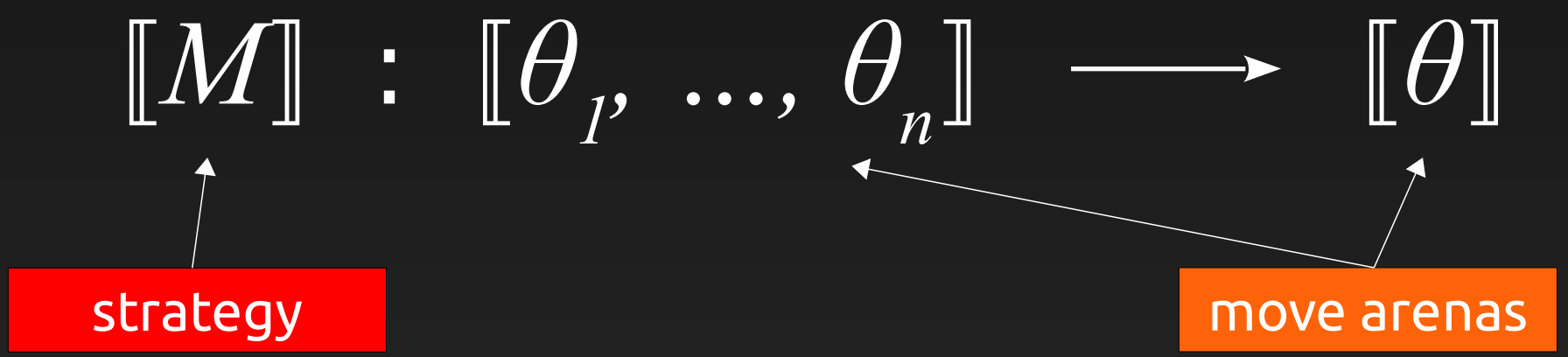
Models cast in nominal sets

- a general theory catering for permutable atomic data
- names: references (ML), objects & threads (Java), etc.

Game anatomy

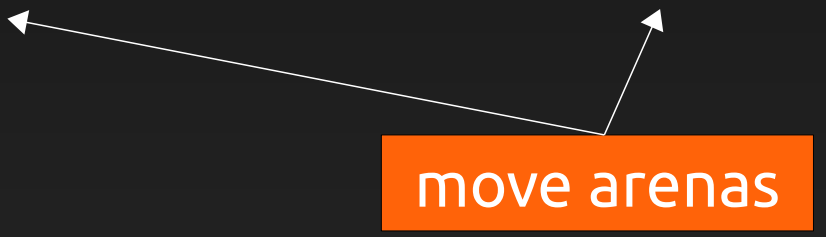


Arenas of moves



Arenas of moves

$$[\![M]\!] : [\![\theta_1, \ldots, \theta_n]\!] \longrightarrow [\![\theta]\!]$$



$$[\![\text{unit}]\!] = \{ * \}$$

$$[\![\text{int}]\!] = \{ 0, 1, -1, \ldots \}$$

$$[\![\text{ref}\ \theta]\!] = \{ a, b, \ldots \}$$

Arenas of moves

$$[\![M]\!] : [\![\theta_1, \ldots, \theta_n]\!] \longrightarrow [\![\theta]\!]$$

move arenas

$\mathcal{N}_\theta$ a set of *names*:
- infinitely many
- comparable for equality only

$$[\![\text{unit}]\!] = \{ * \} = \mathbf{1}$$

$$[\![\text{int}]\!] = \{ 0, 1, -1, \ldots \} = \mathbb{Z}$$

$$[\![\text{ref}\ \theta]\!] = \{ a, b, \ldots \} = \mathcal{N}_\theta$$

Example game

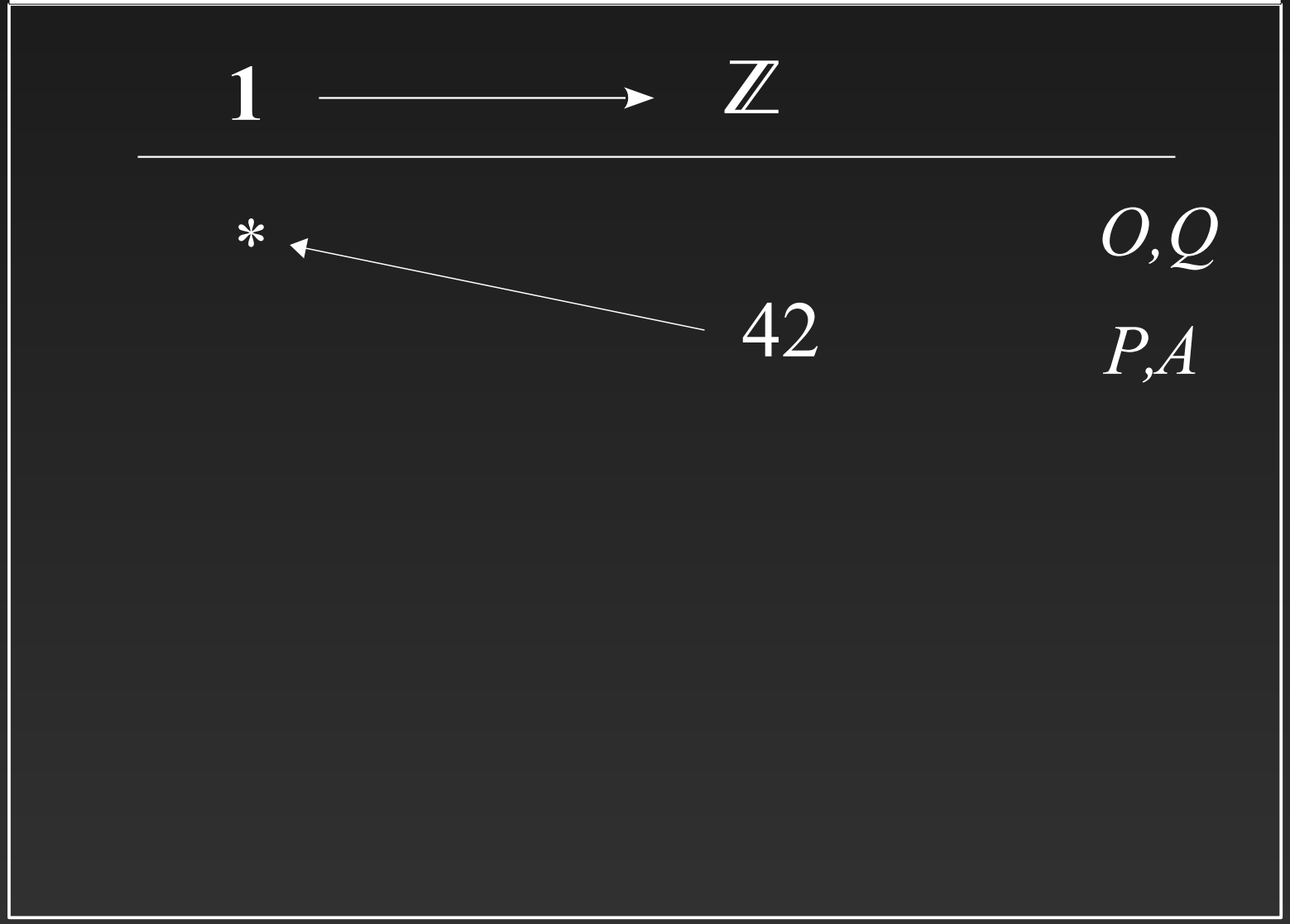
$\vdash 42 : \mathsf{int}$

$\mathbf{1} \longrightarrow \mathbb{Z}$

$*$ $\quad\quad$ O,Q

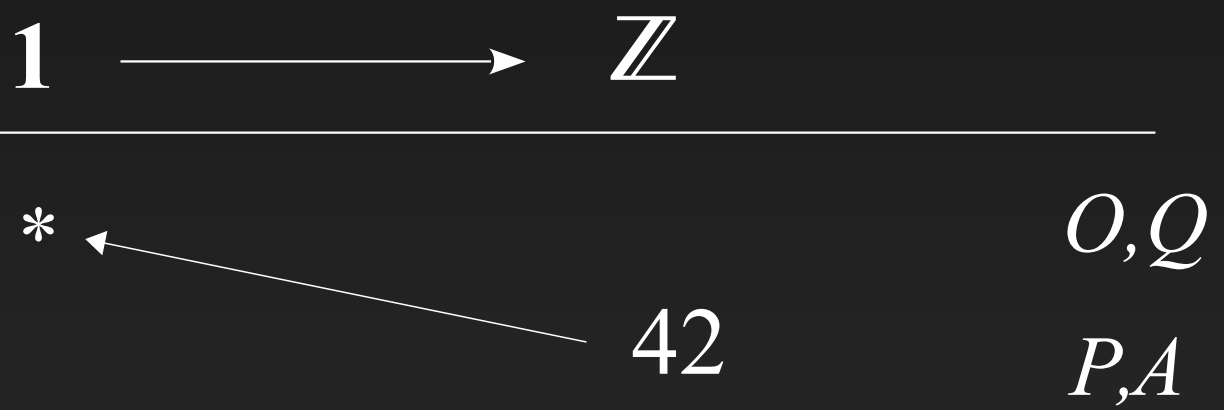
Example game

$\vdash 42 : \mathsf{int}$



Example game

$\vdash 42 : \mathsf{int}$

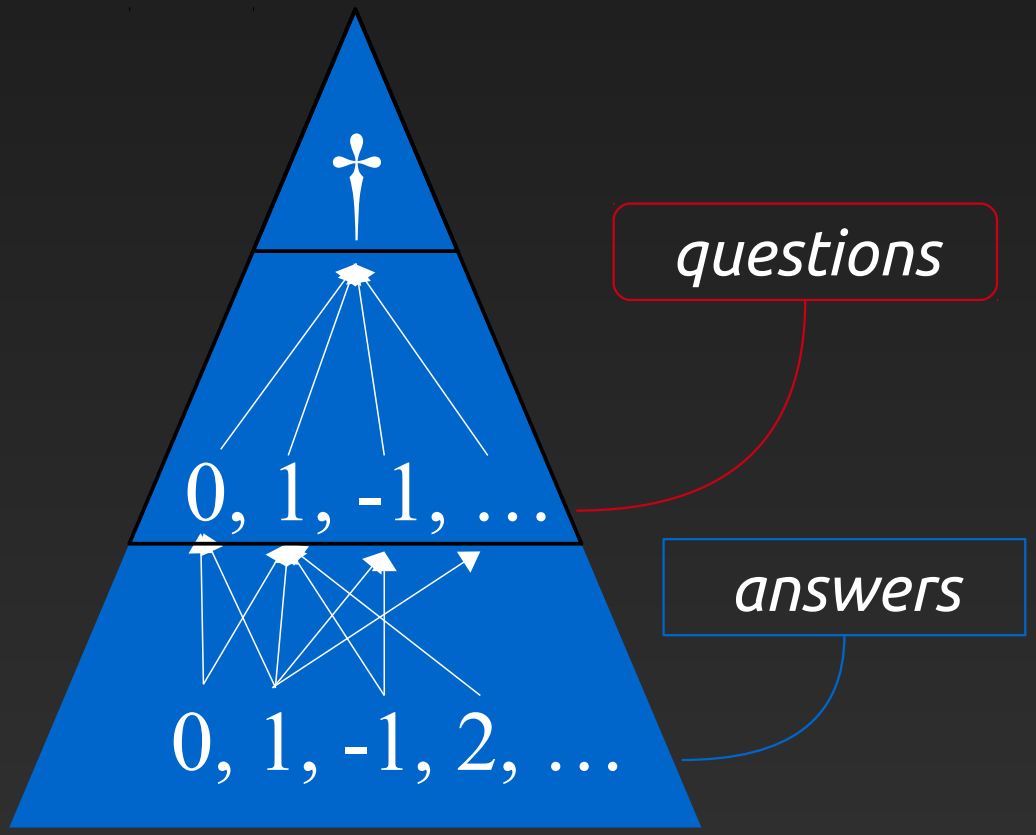


$[\![42]\!] = \{ \; * \;\; 42 \; \}$

$OQ \;\; PA$

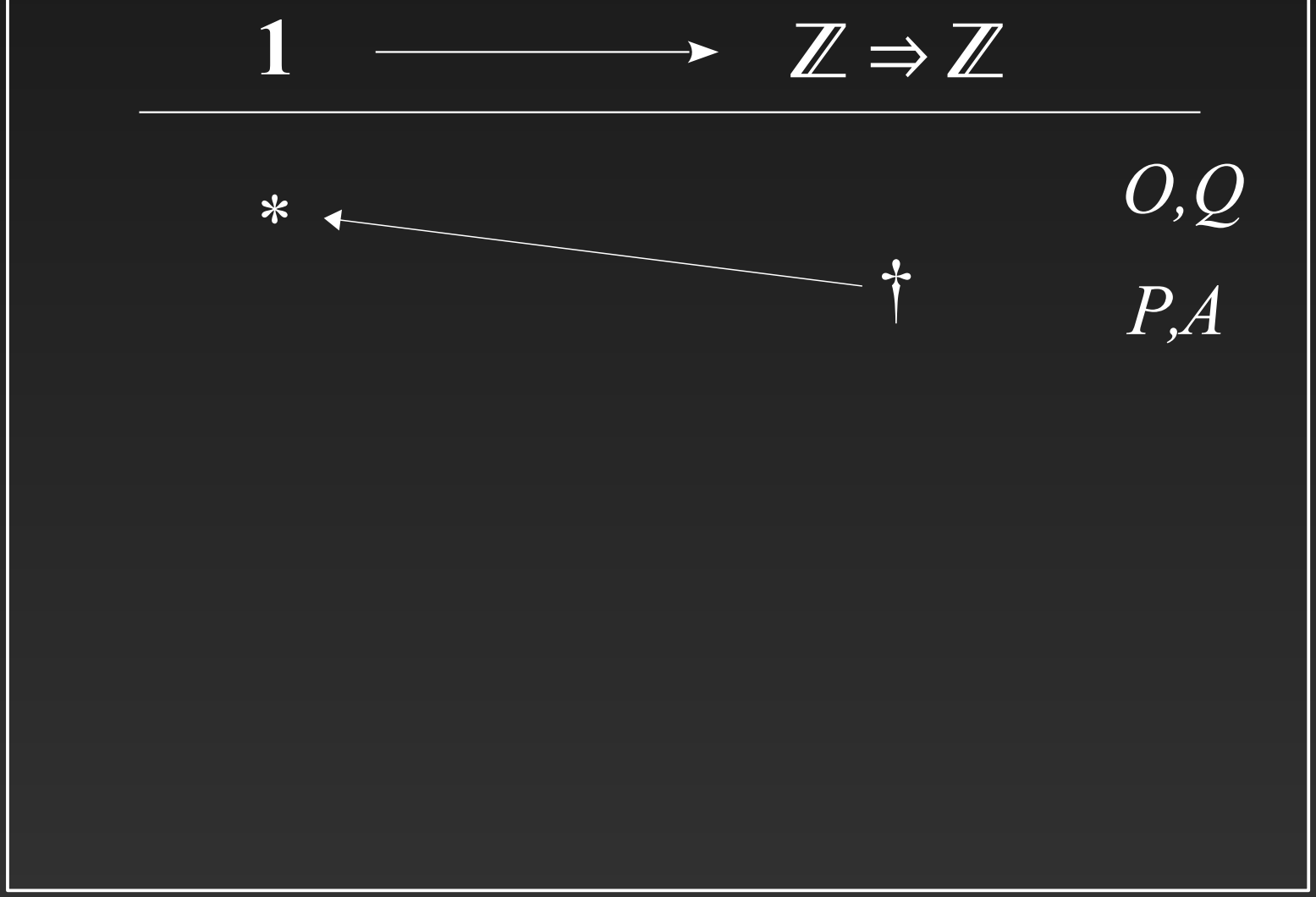
A higher-order arena

$$[\![\text{int} \to \text{int}]\!] = \mathbb{Z} \Rightarrow \mathbb{Z}$$



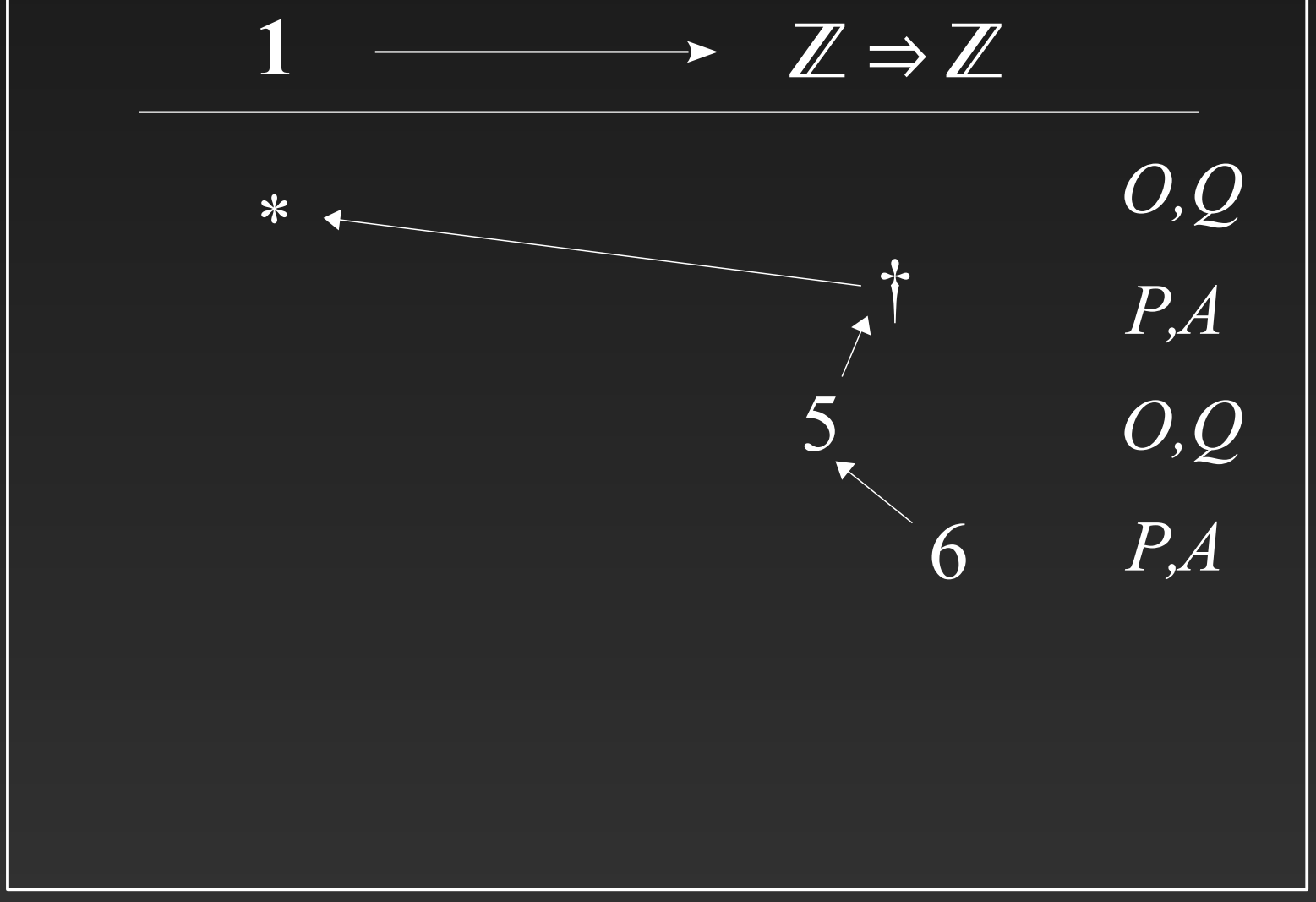
Example game

$\vdash \lambda x.\, x{+}1 : \mathsf{int} \to \mathsf{int}$



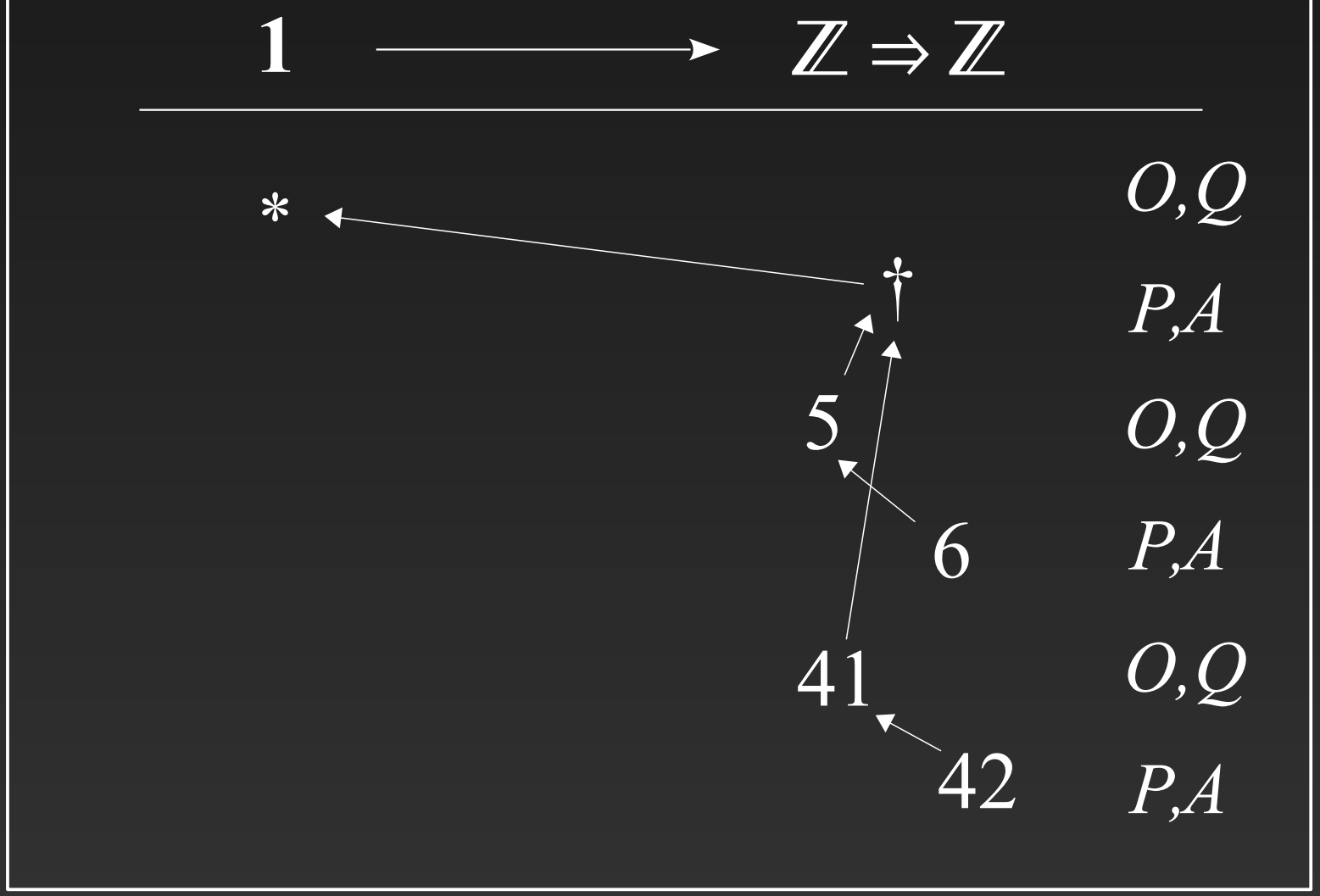
Example game

$\vdash \lambda x.\, x{+}1 : \mathsf{int} \to \mathsf{int}$



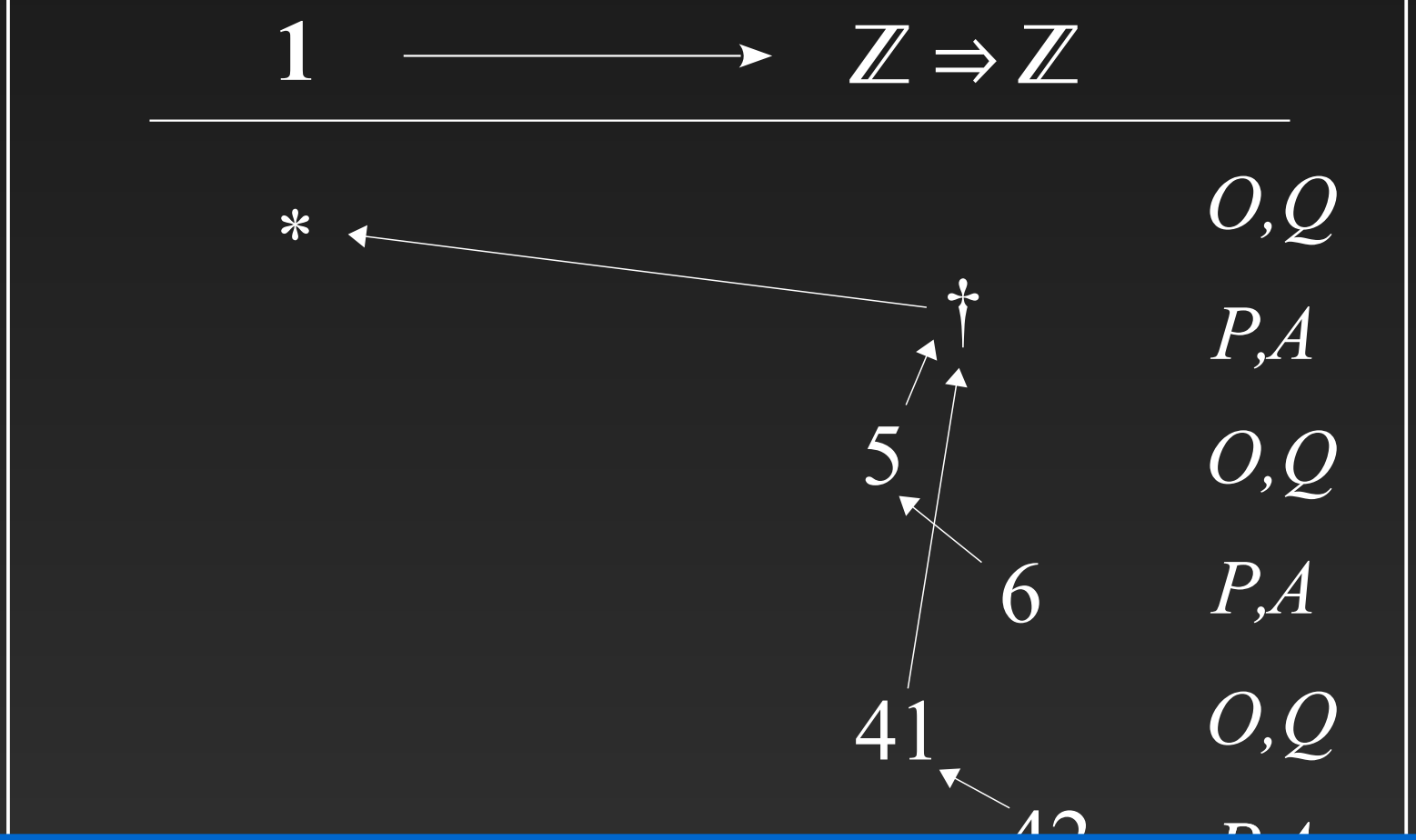
Example game

$\vdash \lambda x.\, x{+}1 : \mathsf{int} \to \mathsf{int}$



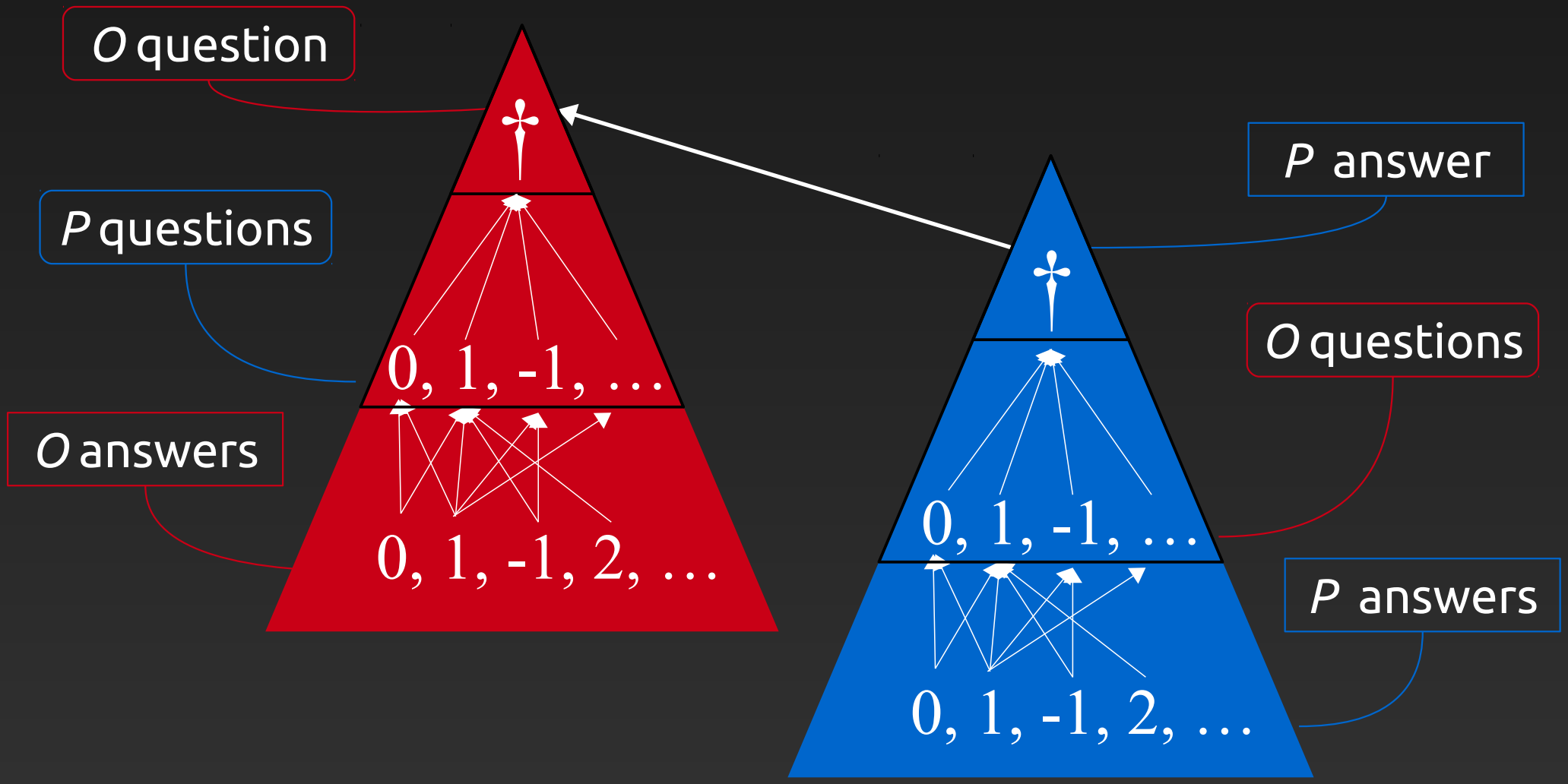
Example game

$\vdash \lambda x.\, x{+}1 : \mathsf{int} \to \mathsf{int}$

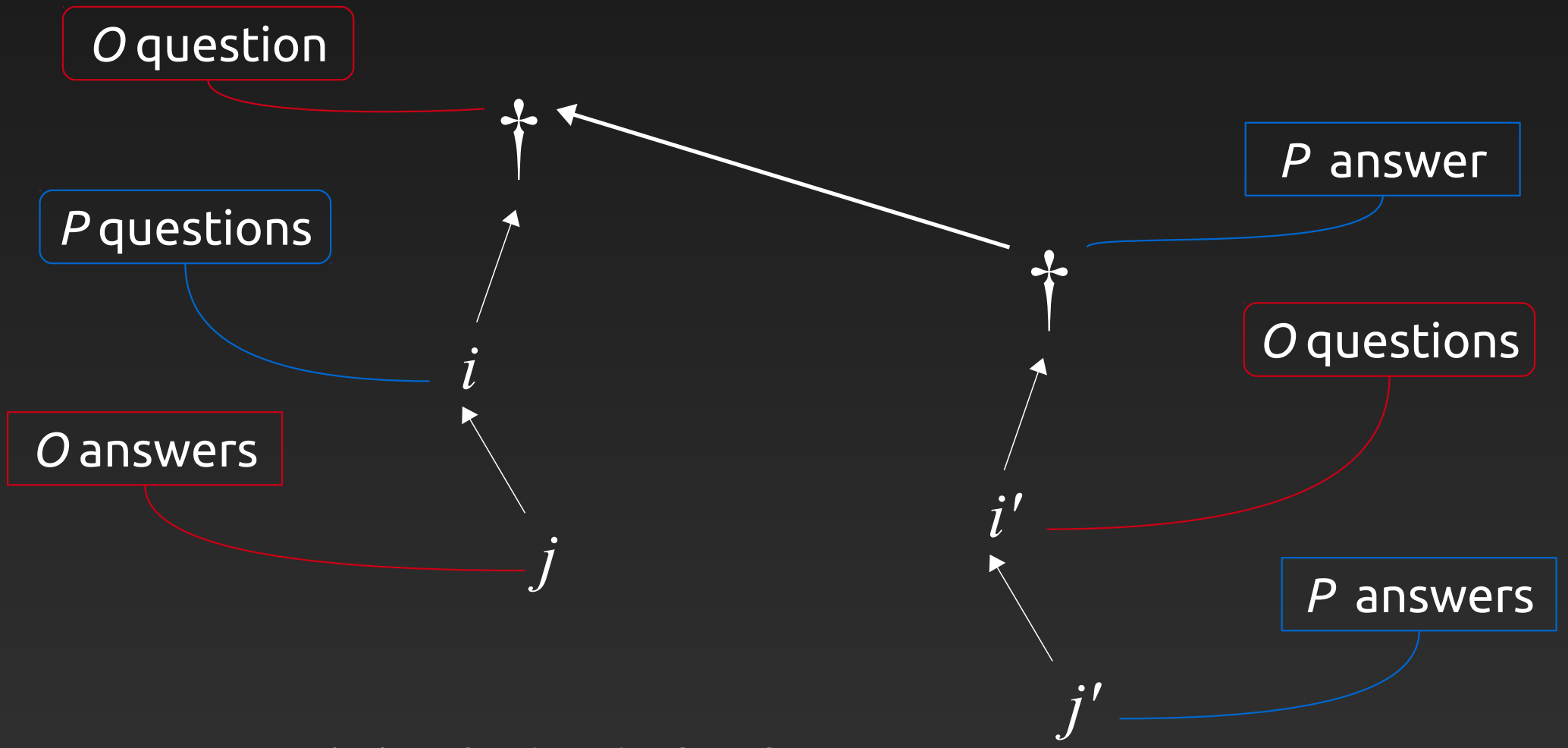


$[\![\lambda x.x{+}1]\!] = \{ \; * \;\; \dagger \;\; i \;\; i{+}1 \; \ldots \; \}$

$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$



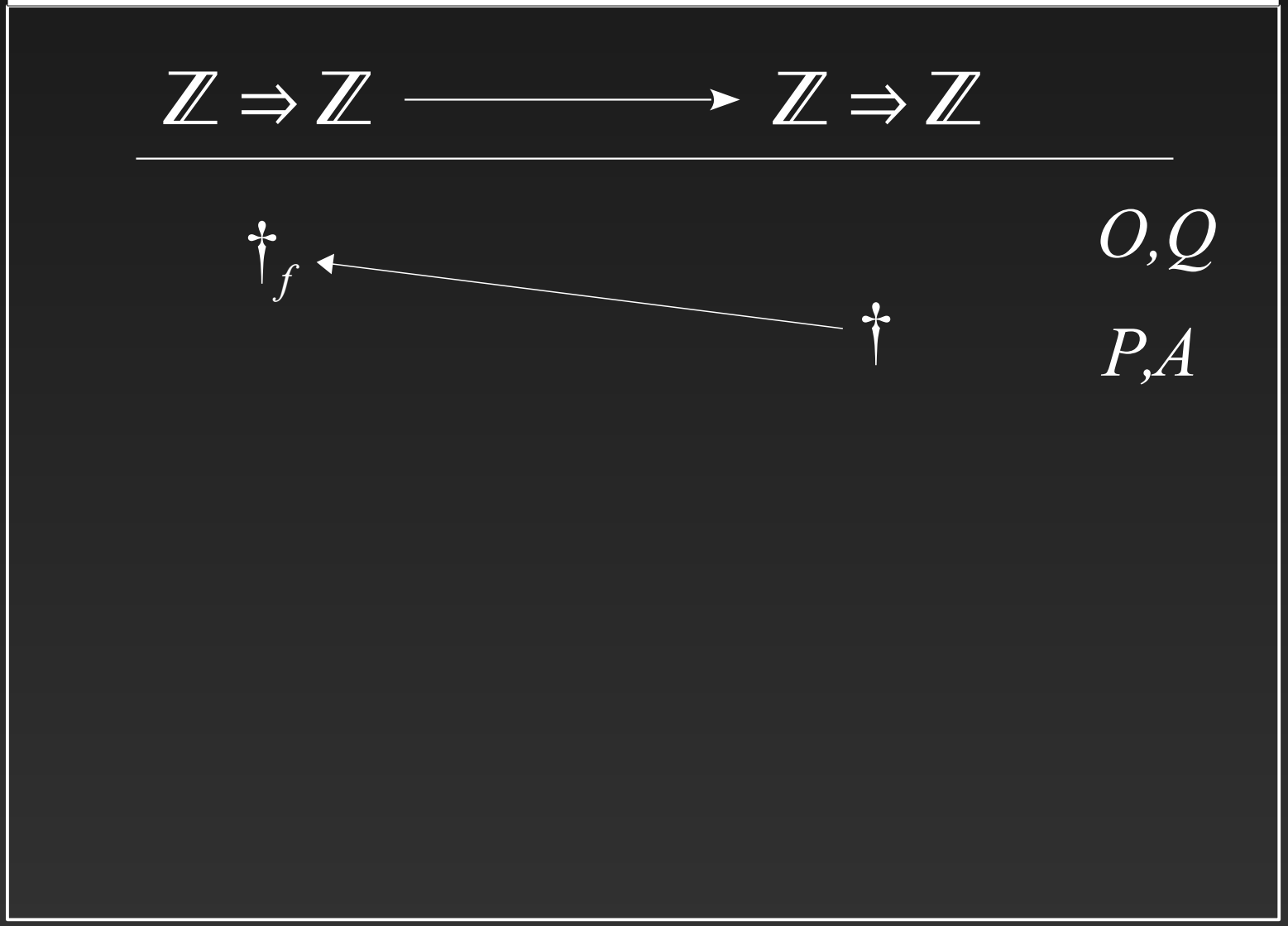
$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$



$i, j, i', j' = 0, 1, -1, 2, -2, \ldots$

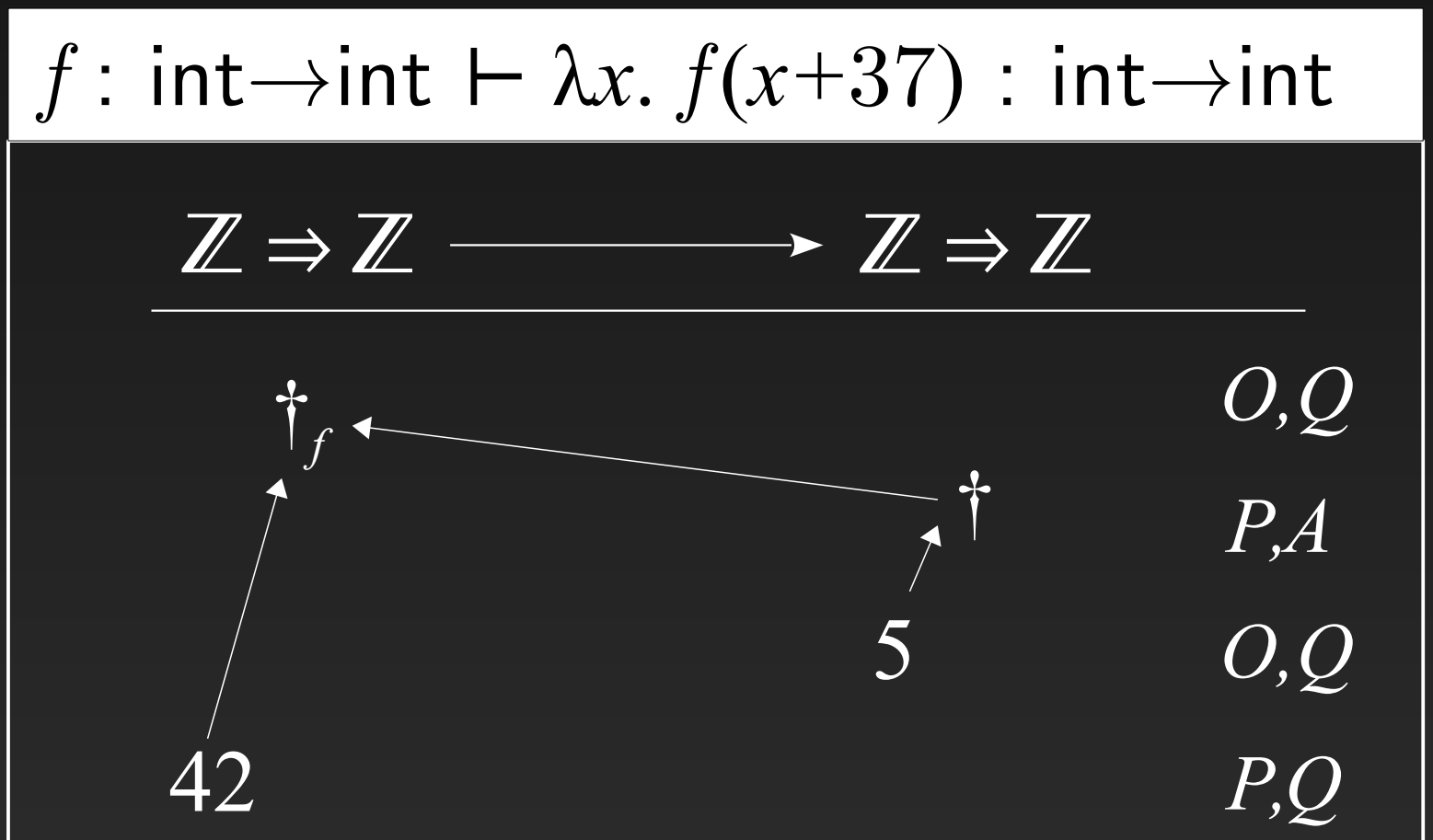
Example game

$f : \mathsf{int} \to \mathsf{int} \vdash \lambda x.\, f(x{+}37) : \mathsf{int} \to \mathsf{int}$



Example game

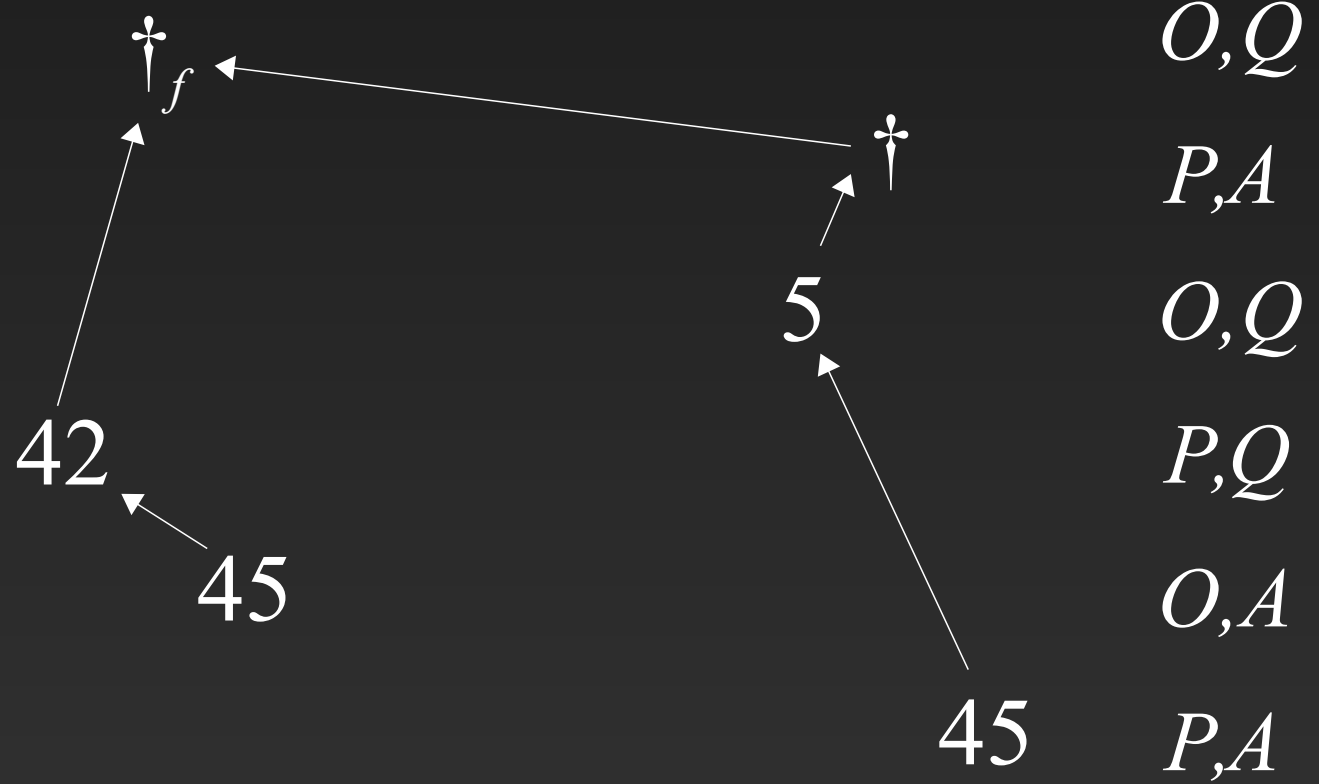
$f : \mathsf{int} \to \mathsf{int} \vdash \lambda x.\, f(x{+}37) : \mathsf{int} \to \mathsf{int}$



Example game

$f : \mathsf{int} \to \mathsf{int} \vdash \lambda x.\, f(x{+}37) : \mathsf{int} \to \mathsf{int}$

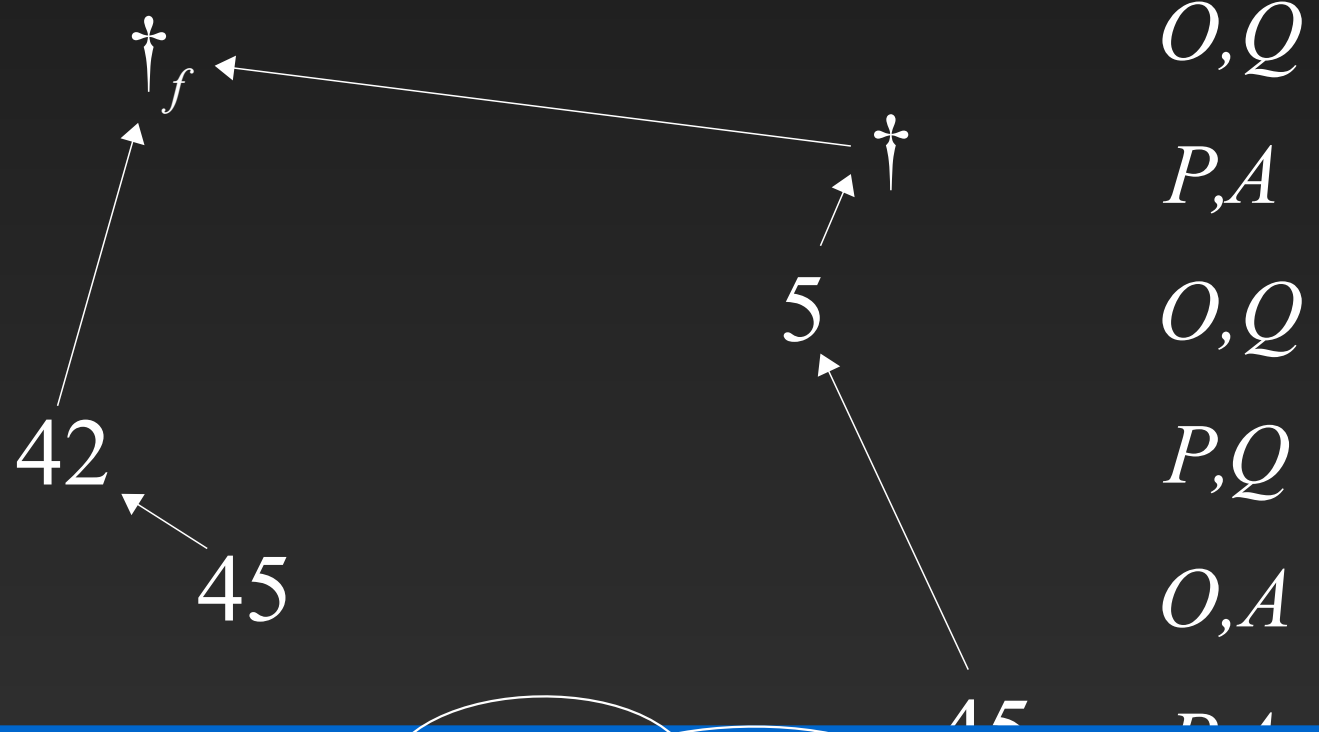
$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$



Example game

$f : \mathsf{int} \to \mathsf{int} \vdash \lambda x.\, f(x{+}37) : \mathsf{int} \to \mathsf{int}$

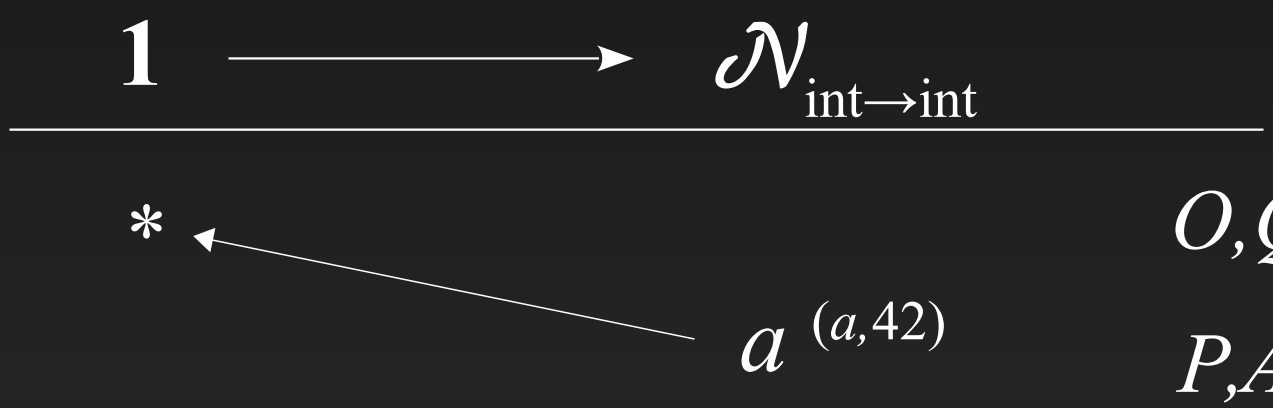
$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$



$[\![\lambda x.f(x{+}37)]\!] = \{\ \dagger_f\ \dagger\ i\ (i{+}37)\ j\ j\ \ldots\ \}$

Nominal games

$\vdash \mathsf{ref}\ 42 : \mathsf{ref\ int}$

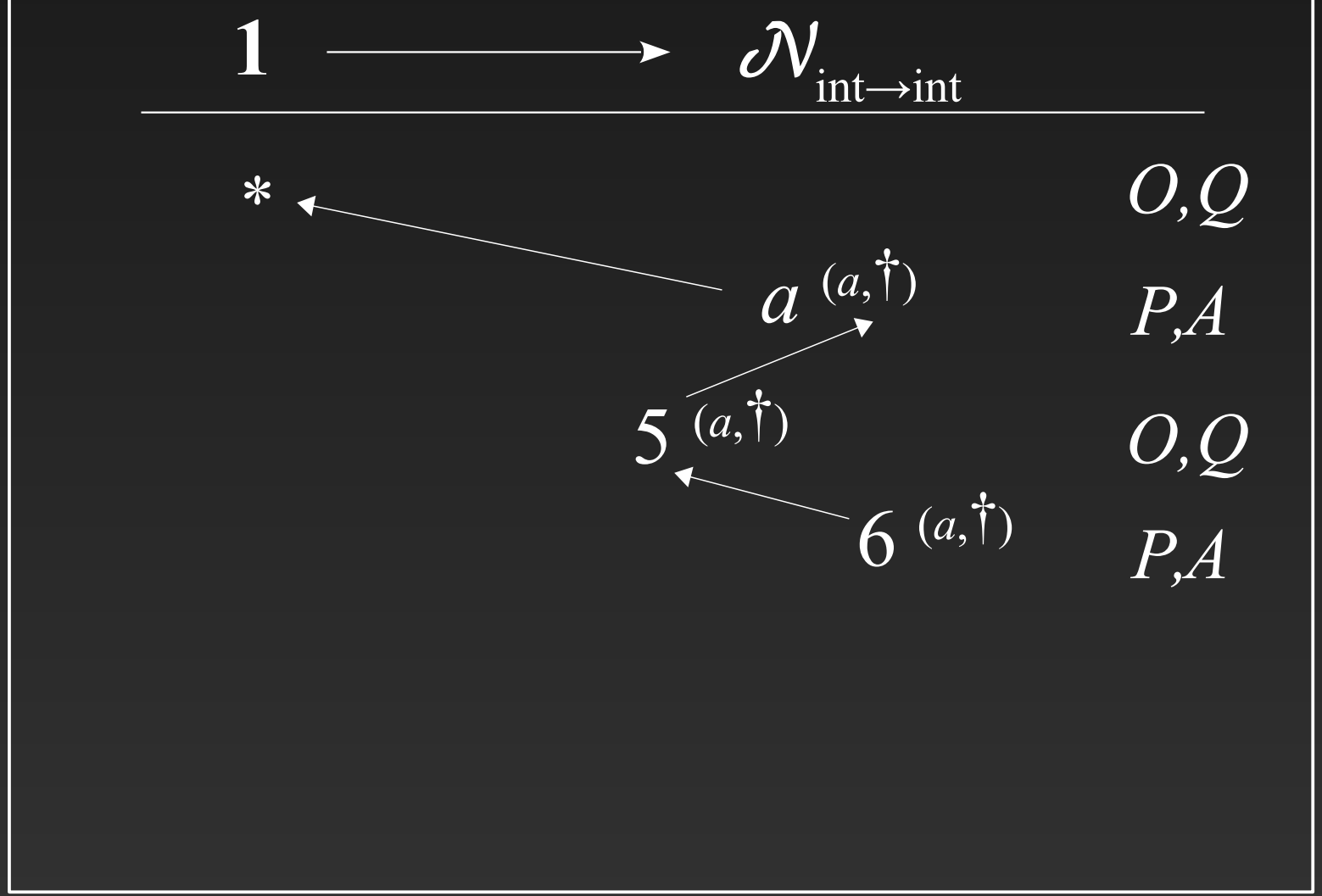


$[\![\mathsf{ref}\ 42]\!] = \{\ * \quad a^{(a,42)}\ \}$

$OQ \quad PA$

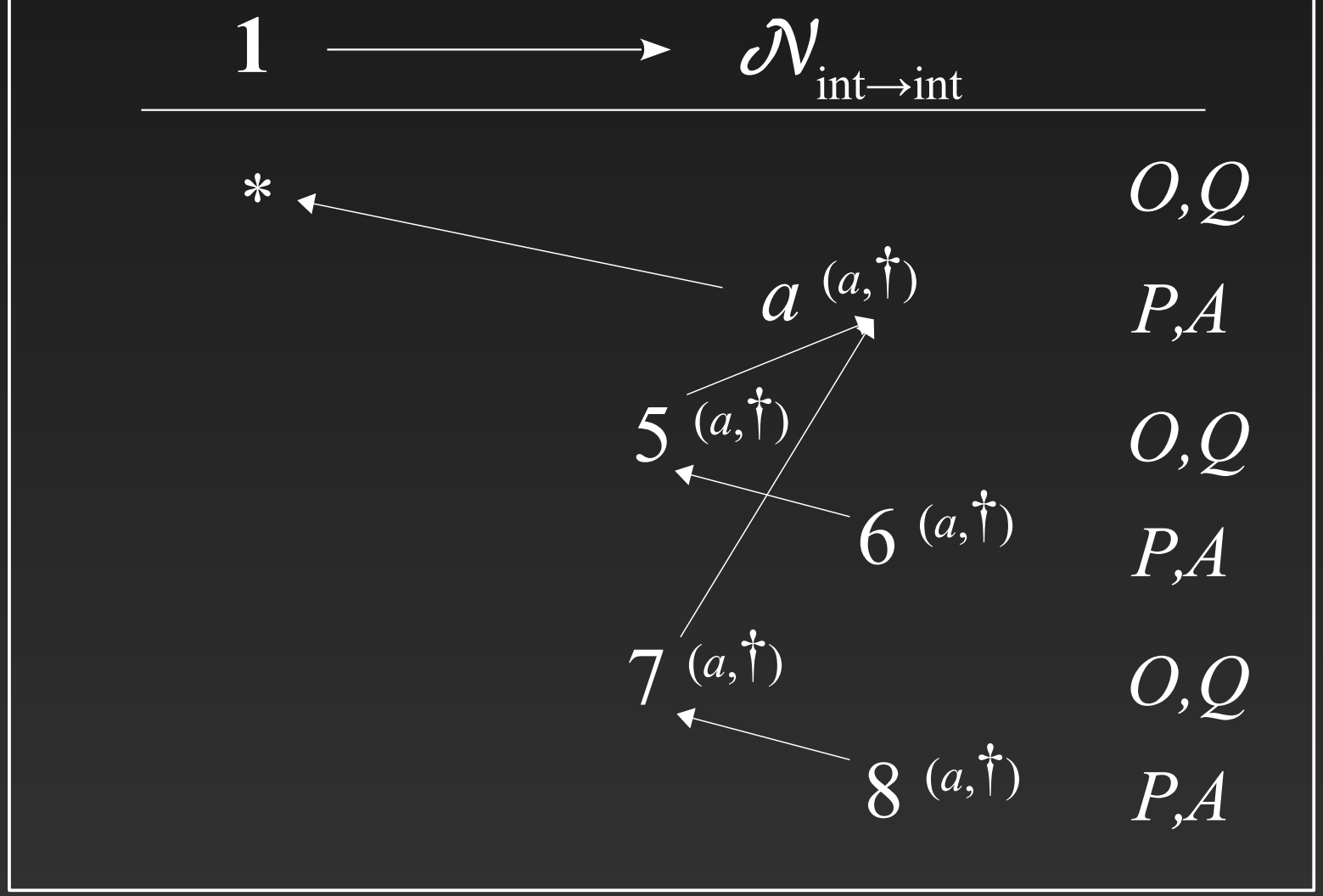
Nominal games

$\vdash \mathsf{ref}\,(\lambda x^{\mathsf{int}}.x{+}1) : \mathsf{ref}\,(\mathsf{int}{\to}\mathsf{int})$



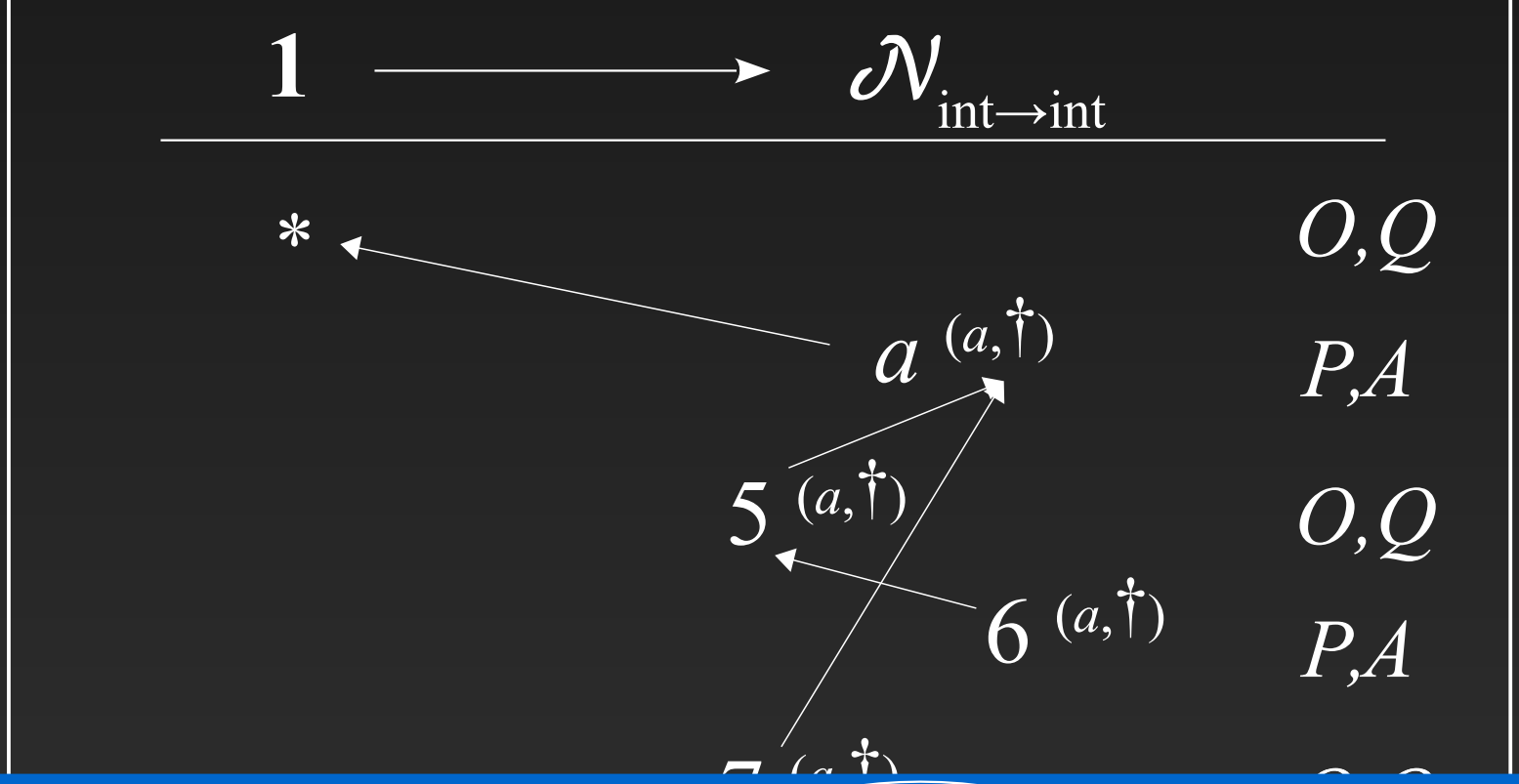
Nominal games

$\vdash \mathsf{ref}\,(\lambda x^{\mathsf{int}}.x{+}1) : \mathsf{ref}\,(\mathsf{int}{\to}\mathsf{int})$



Nominal games

$\vdash \mathsf{ref}\ (\lambda x^{\mathsf{int}}.x{+}1) : \mathsf{ref}\,(\mathsf{int}{\to}\mathsf{int})$

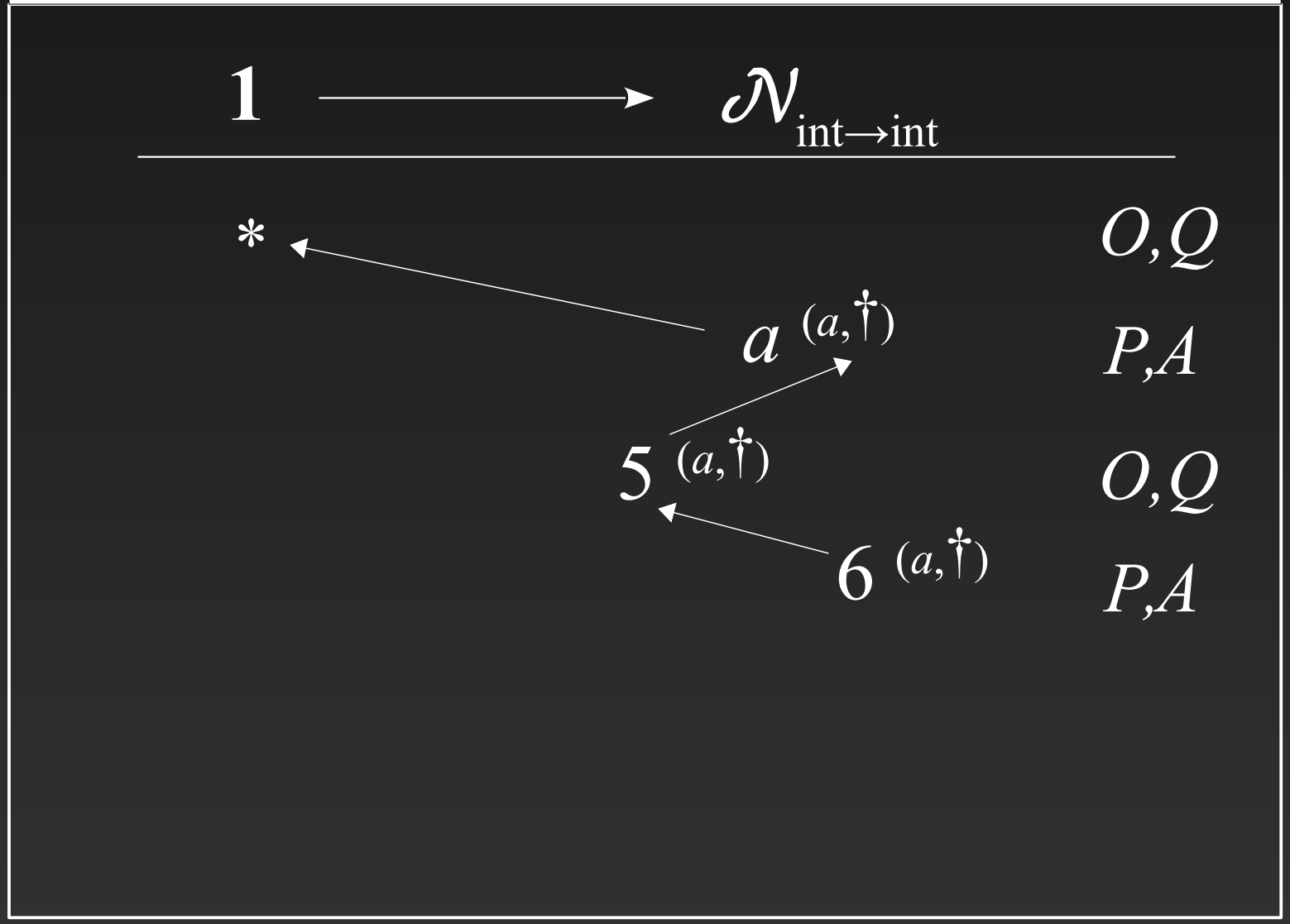


$[\![\mathsf{ref}\ \lambda x.x{+}1]\!] = \{\ *\ \ a^{(a,\dagger)}\ \ 5^{(a,\dagger)}\ \ 6^{(a,\dagger)}\ \ 7^{(a,\dagger)}\ \ 8^{(a,\dagger)}\ \ \ldots\ \}$

$OQ\ \ PA\ \ \ OQ\ \ \ PA\ \ \ OQ\ \ \ PA$

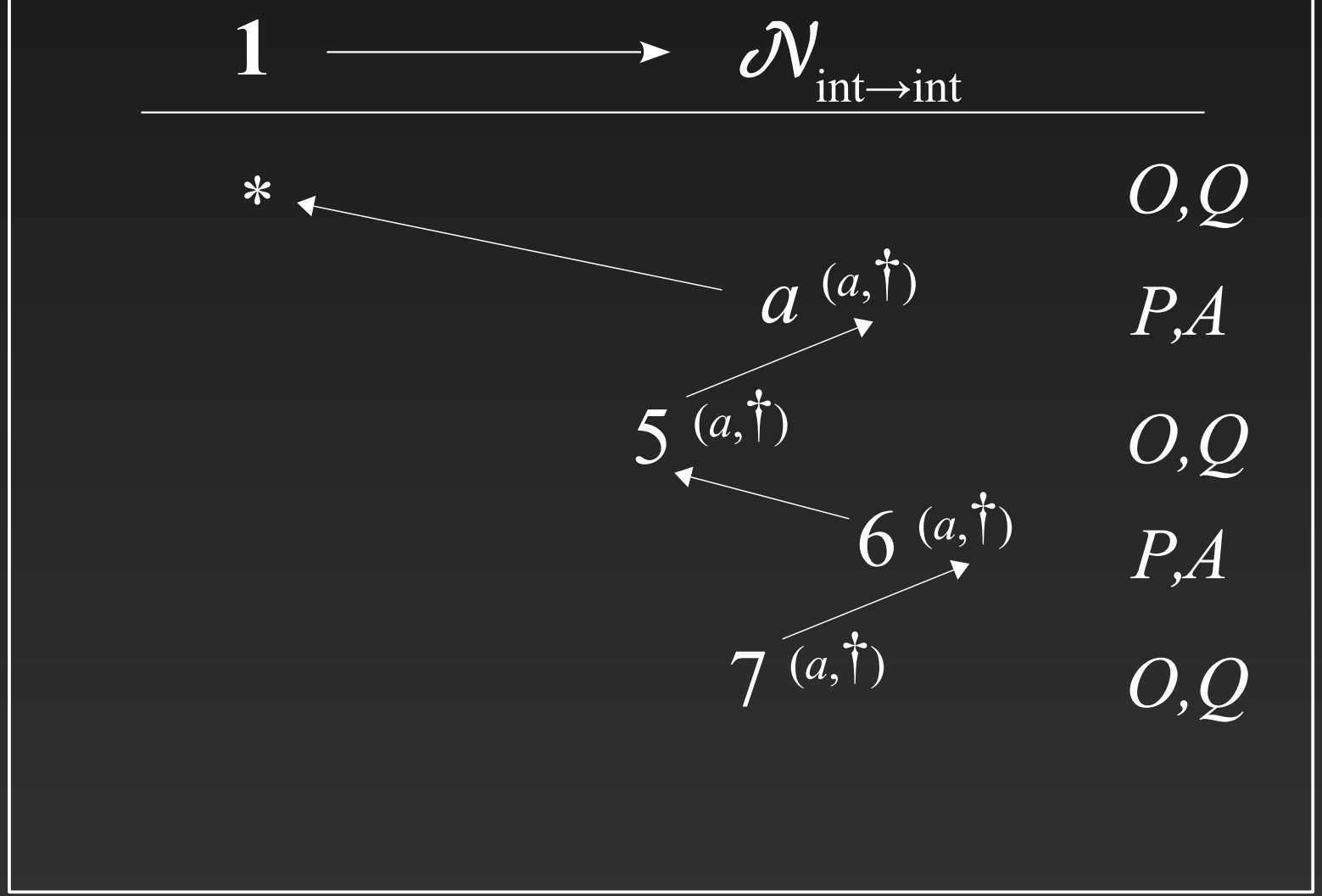
(there are more plays)

$\vdash \mathsf{ref}\,(\lambda x^{\mathsf{int}}.x+1) : \mathsf{ref}\,(\mathsf{int}\to\mathsf{int})$



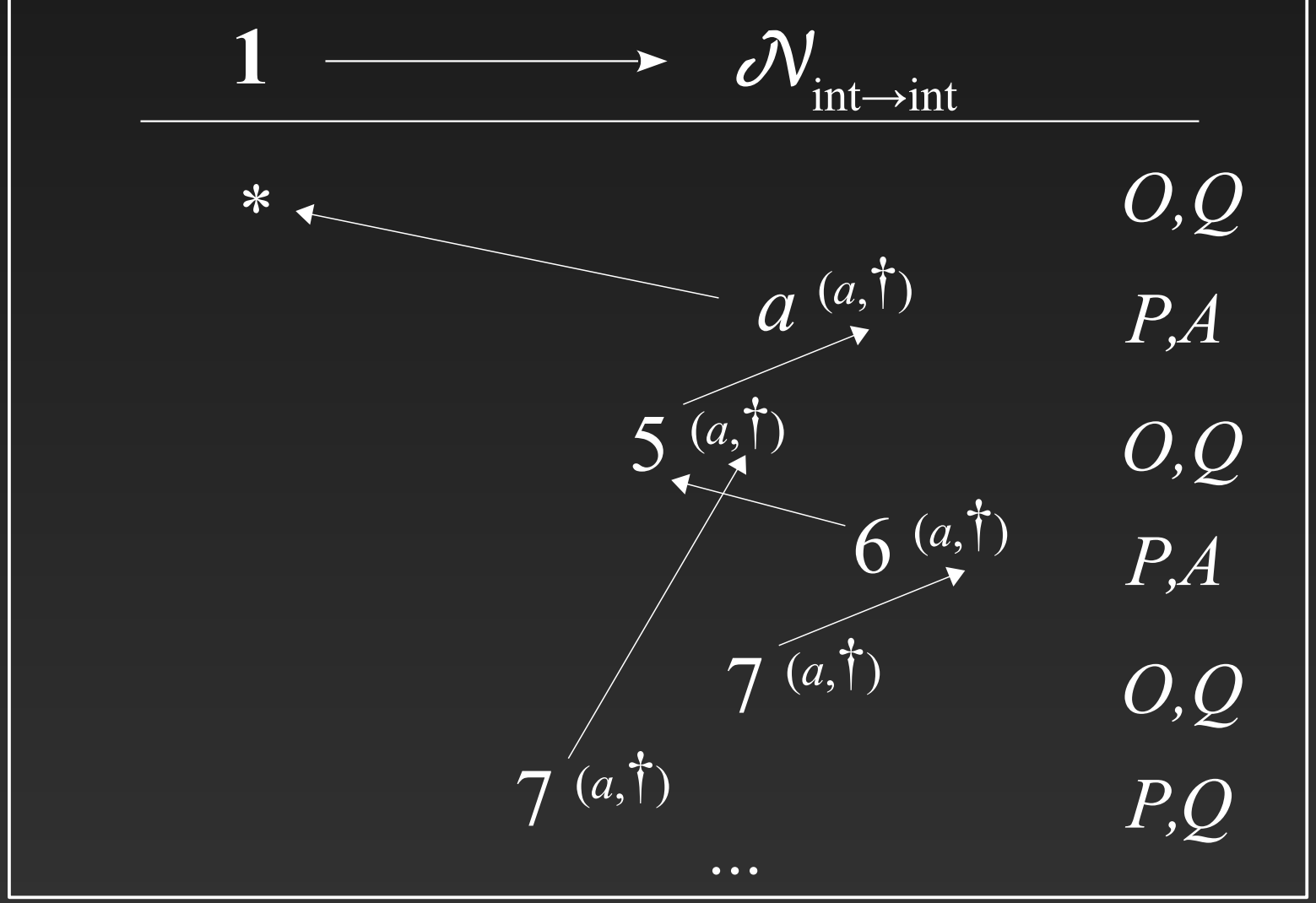
(there are more plays)

$\vdash \mathsf{ref}\,(\lambda x^{\mathsf{int}}.x{+}1) : \mathsf{ref}\,(\mathsf{int}{\to}\mathsf{int})$



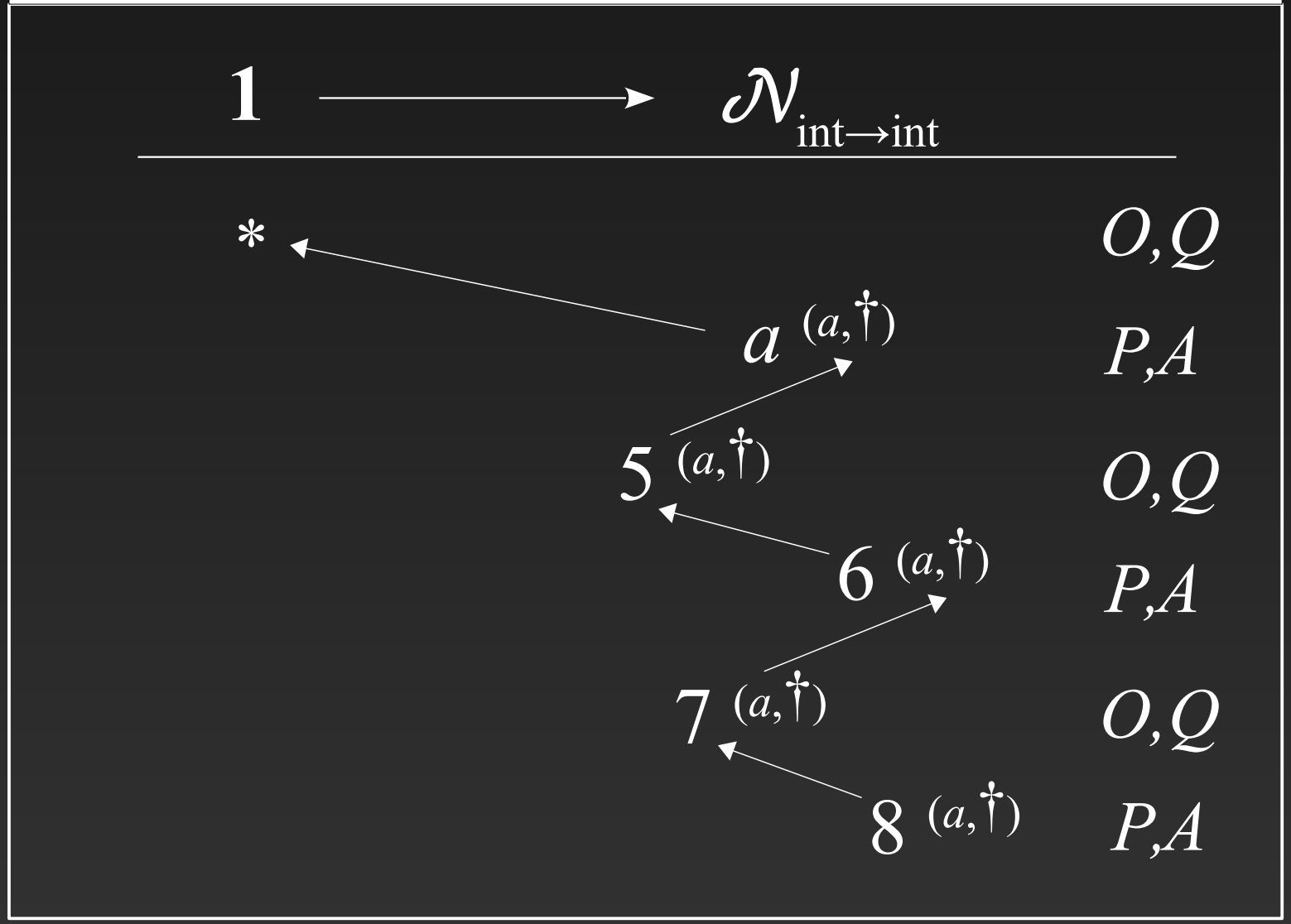
(there are more plays)

$\vdash \mathsf{ref}\,(\lambda x^{\mathsf{int}}.x+1) : \mathsf{ref}\,(\mathsf{int}\to\mathsf{int})$



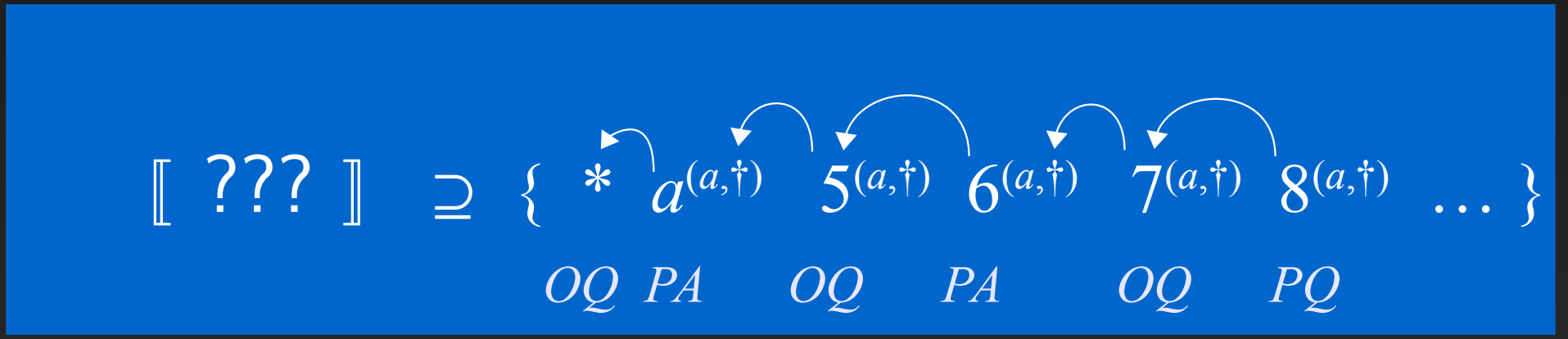
Quiz

$\vdash$??? : ref (int$\rightarrow$int)



Quiz

$\vdash$??? : ref (int$\rightarrow$int)



Quiz

$\vdash$??? : ref (int$\to$int)

$[\![\text{???}]\!] \supseteq \{ * \; a^{(a,\dagger)} \; 5^{(a,\dagger)} \; 6^{(a,\dagger)} \; 7^{(a,\dagger)} \; 8^{(a,\dagger)} \; \ldots \}$

$OQ \;\; PA \quad OQ \quad PA \quad OQ \quad PQ$

$\mathit{let\ rec}\ l = \mathsf{ref}\ (\lambda x^{\mathsf{int}}.(l := \lambda y^{\mathsf{int}}.y{+}1);\ x{+}1);\ l$

i.e.

$\mathsf{let}\ l = \mathsf{ref}\ \lambda x^{\mathsf{int}}.0\ \ \mathsf{in}$

$\quad (l := \lambda x^{\mathsf{int}}.(l := \lambda y^{\mathsf{int}}.y{+}1);\ x{+}1);\ l\ \ : \mathsf{ref}(\mathsf{int}\to\mathsf{int})$

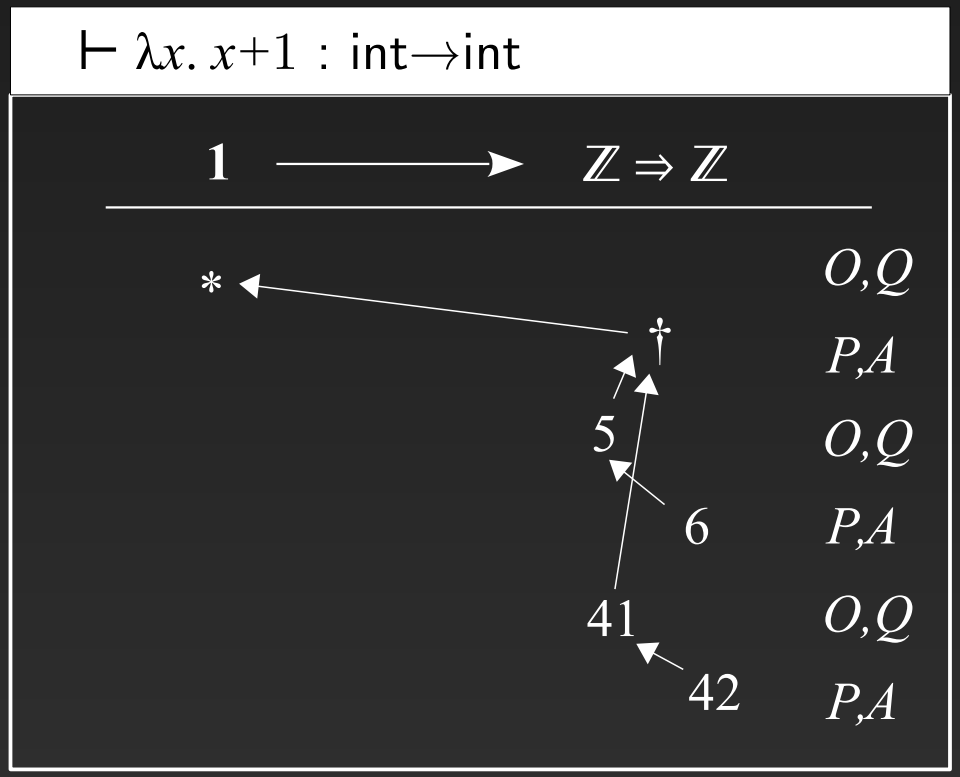
Composition

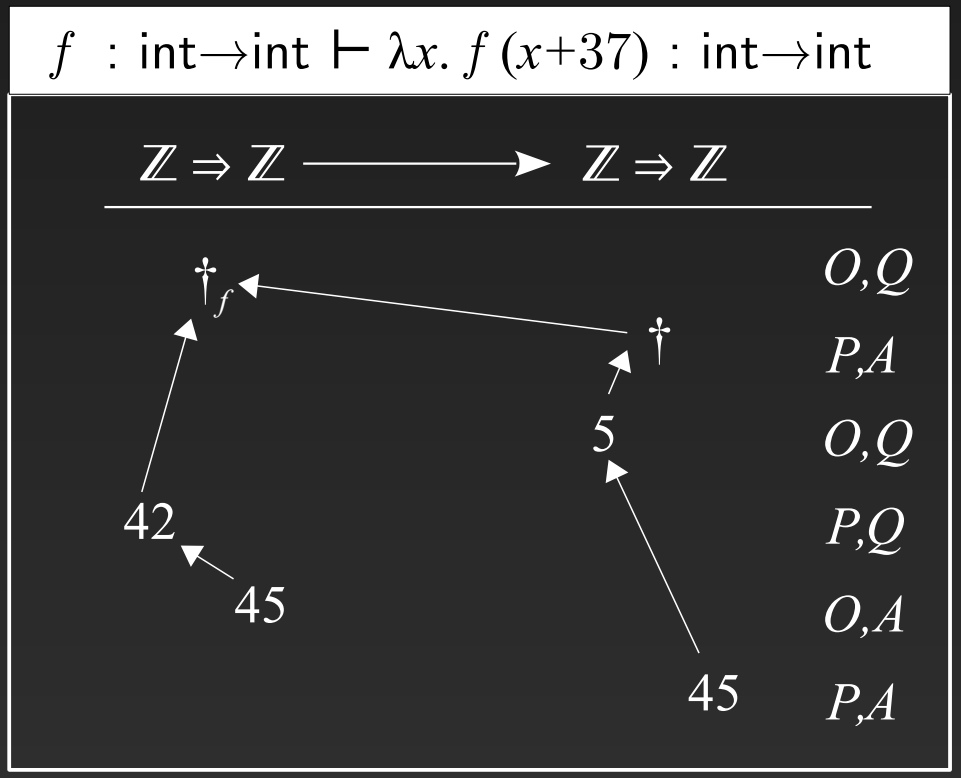
Strategies can be composed by matching O/P behaviours

Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int

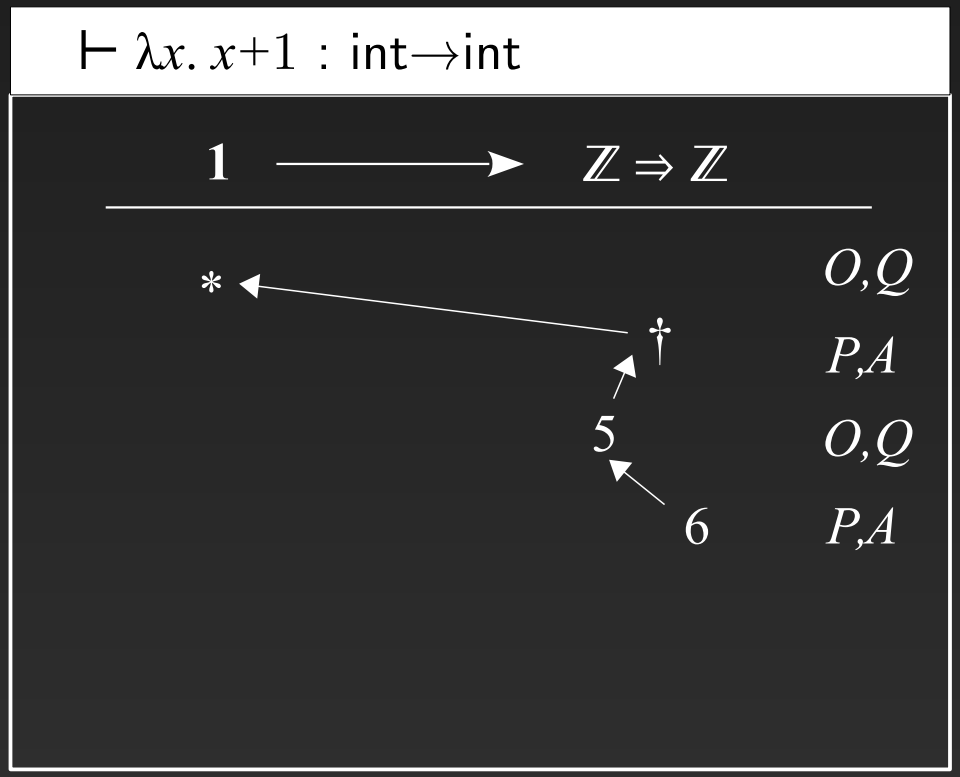


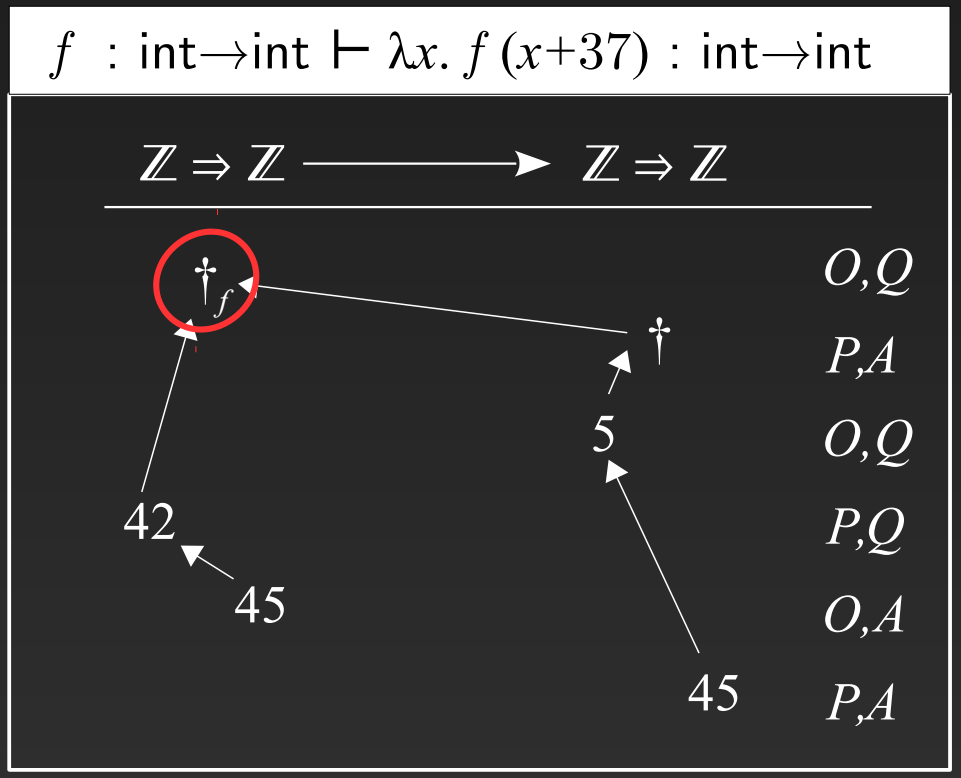


Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int

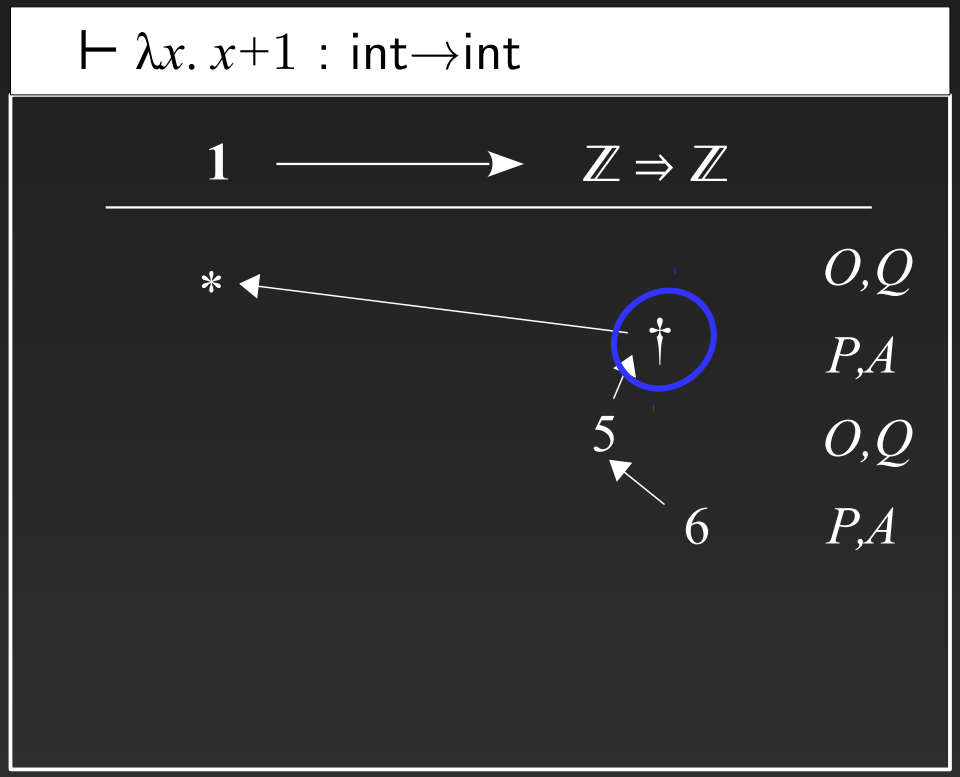


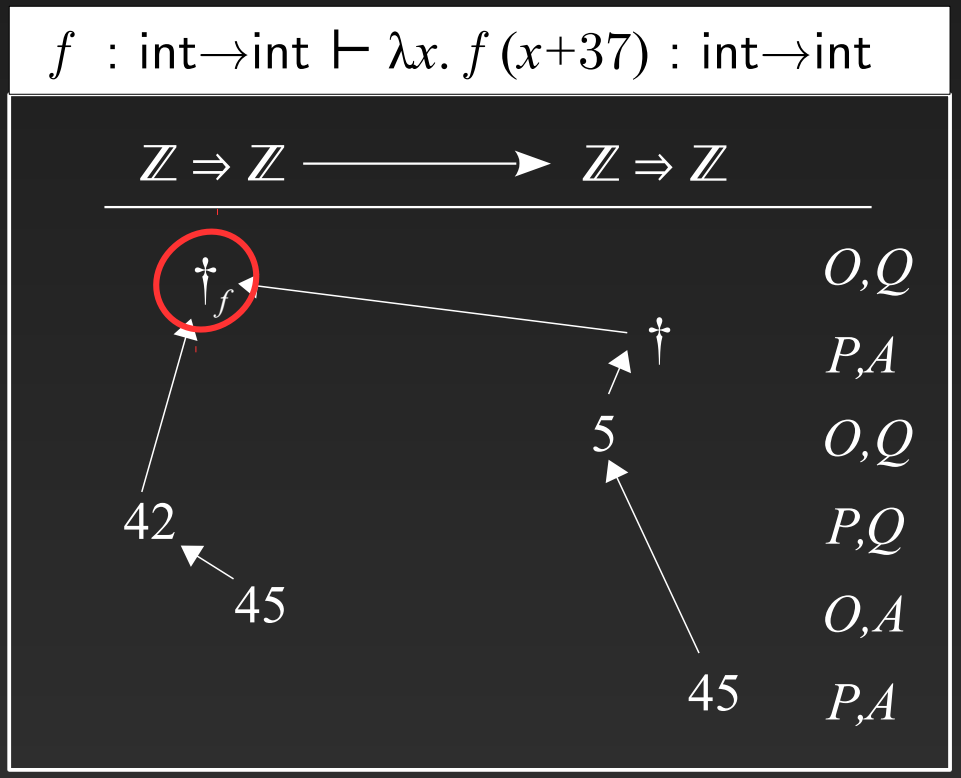


Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int

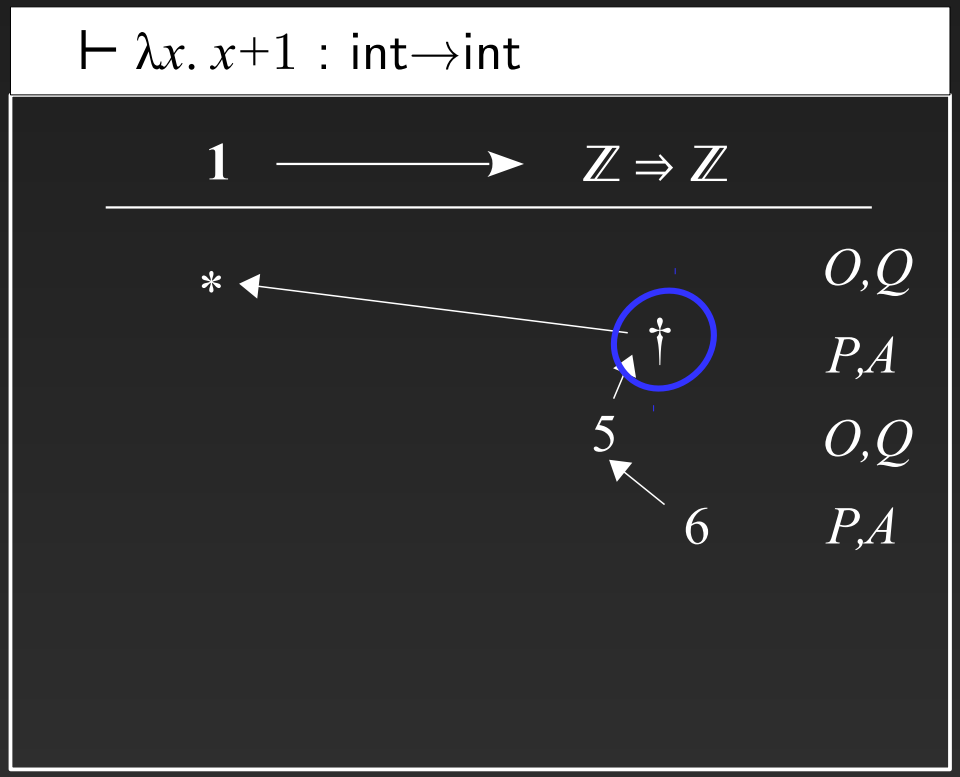


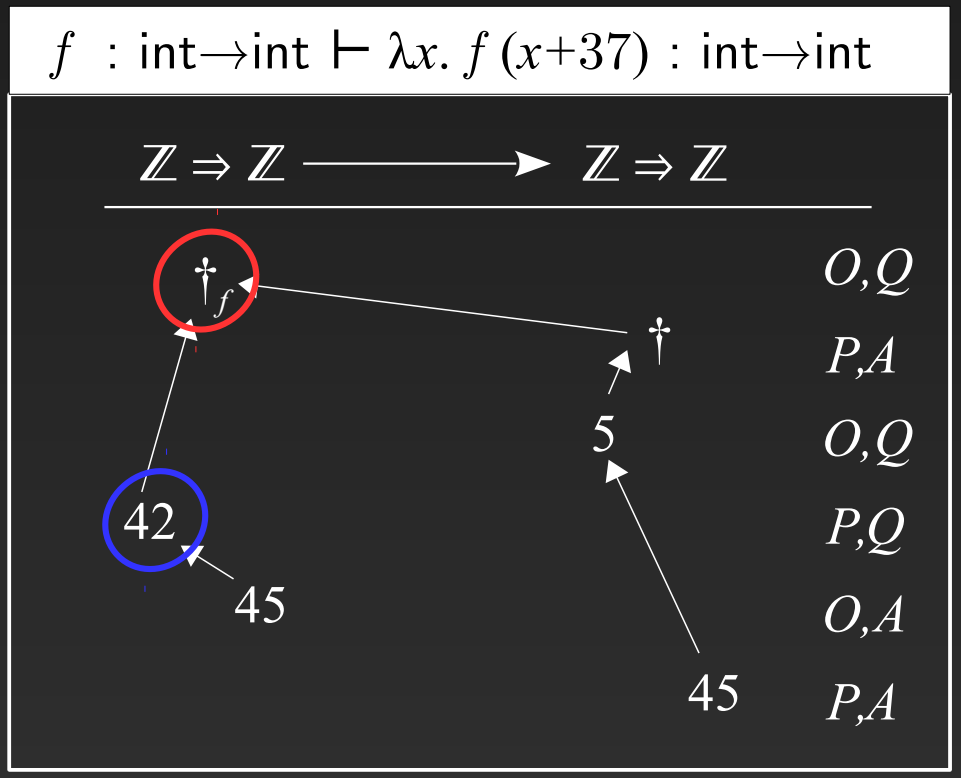


Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int

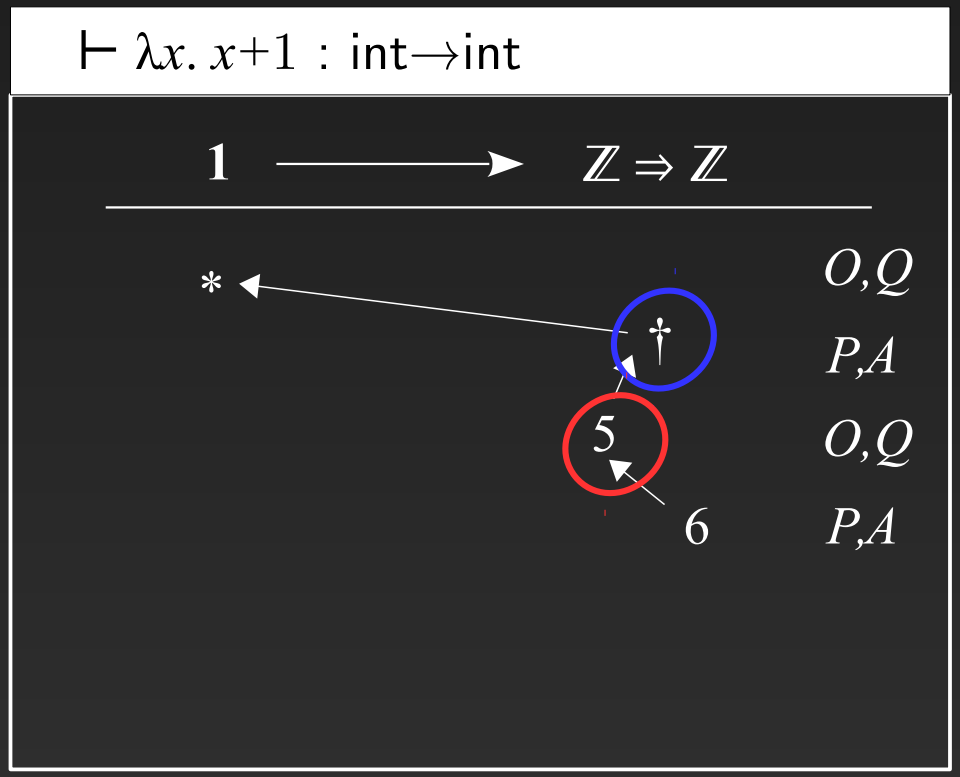


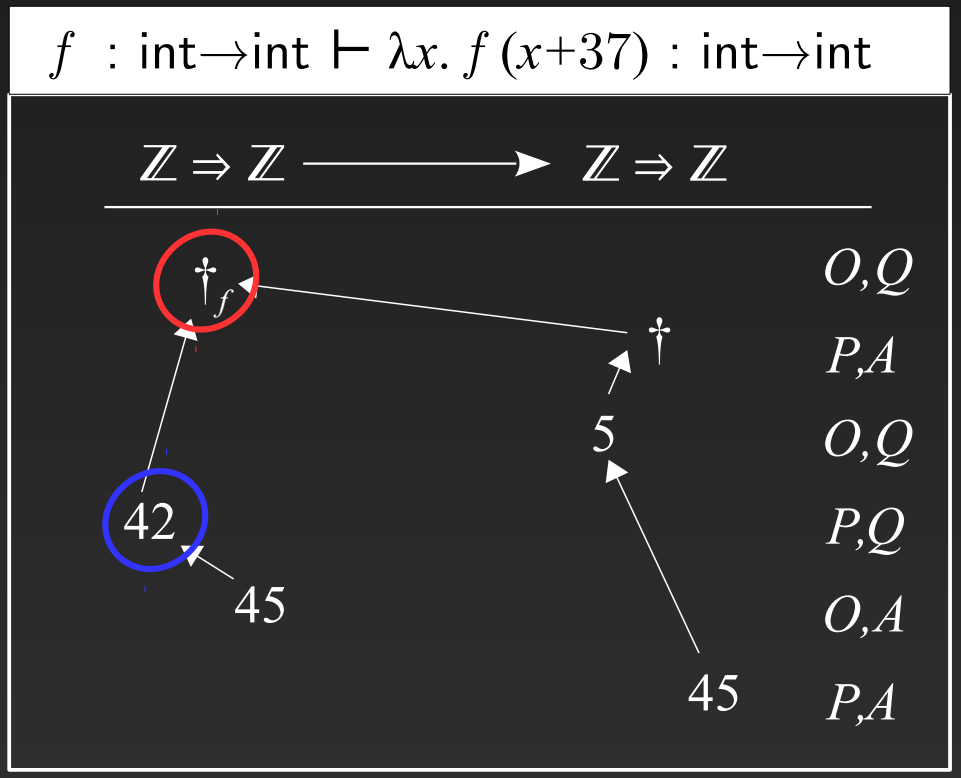


Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int

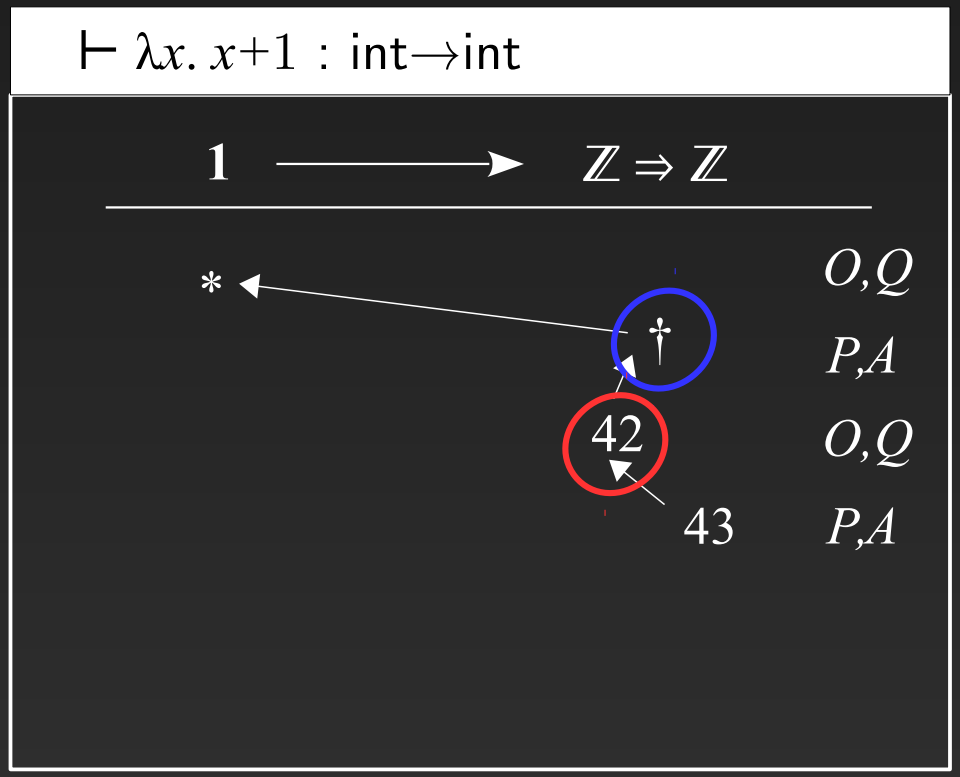


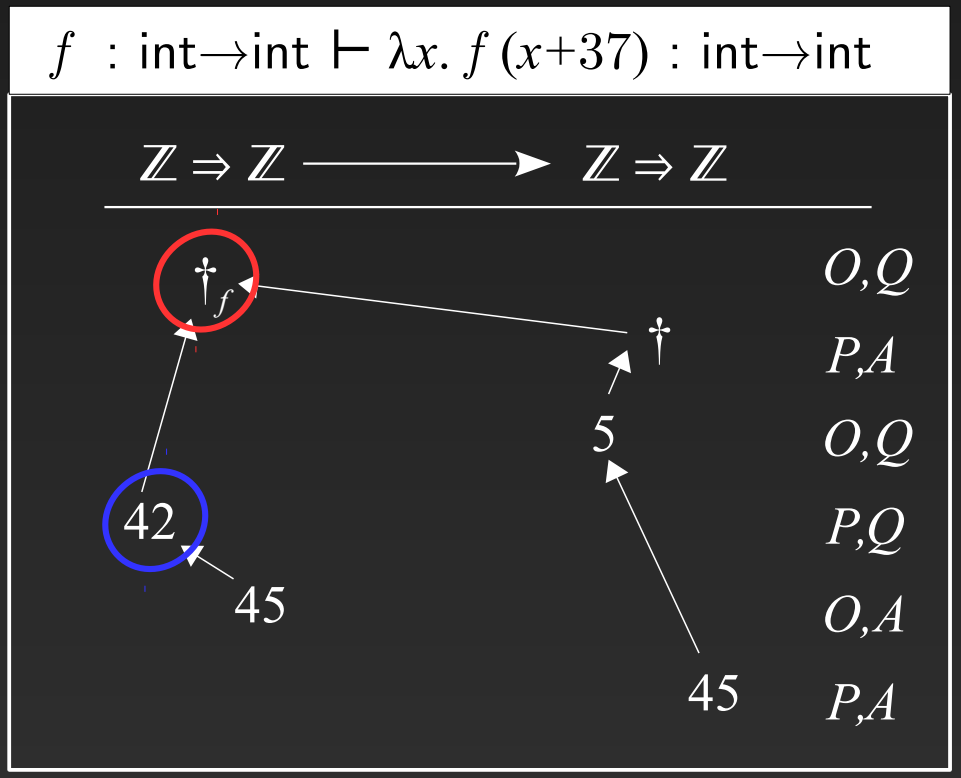


Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int

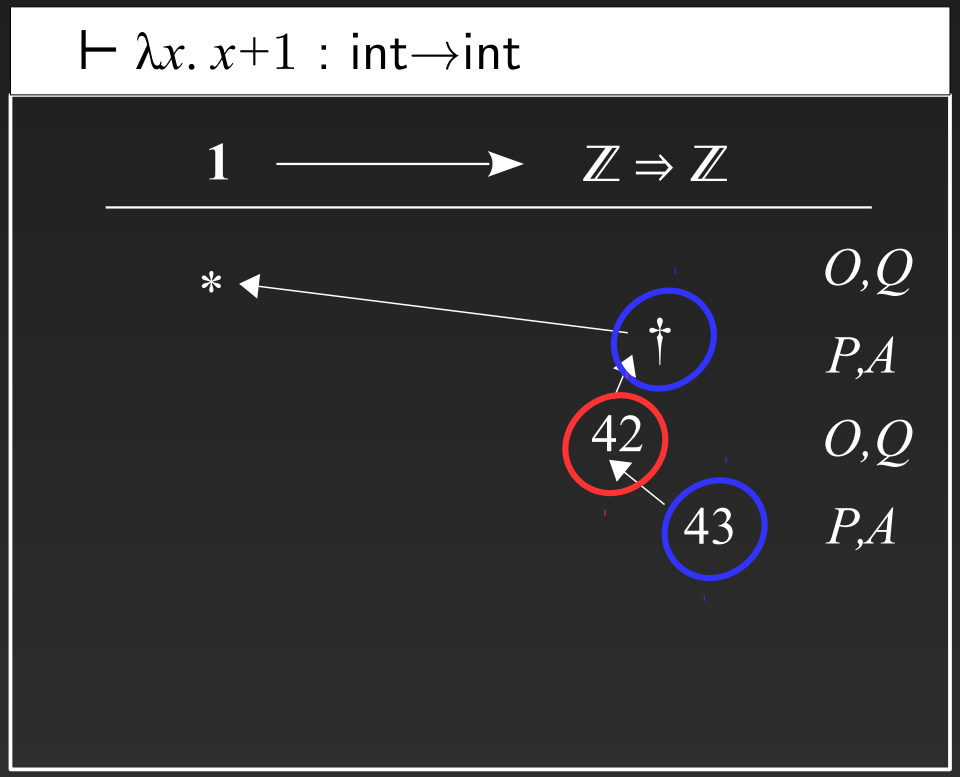


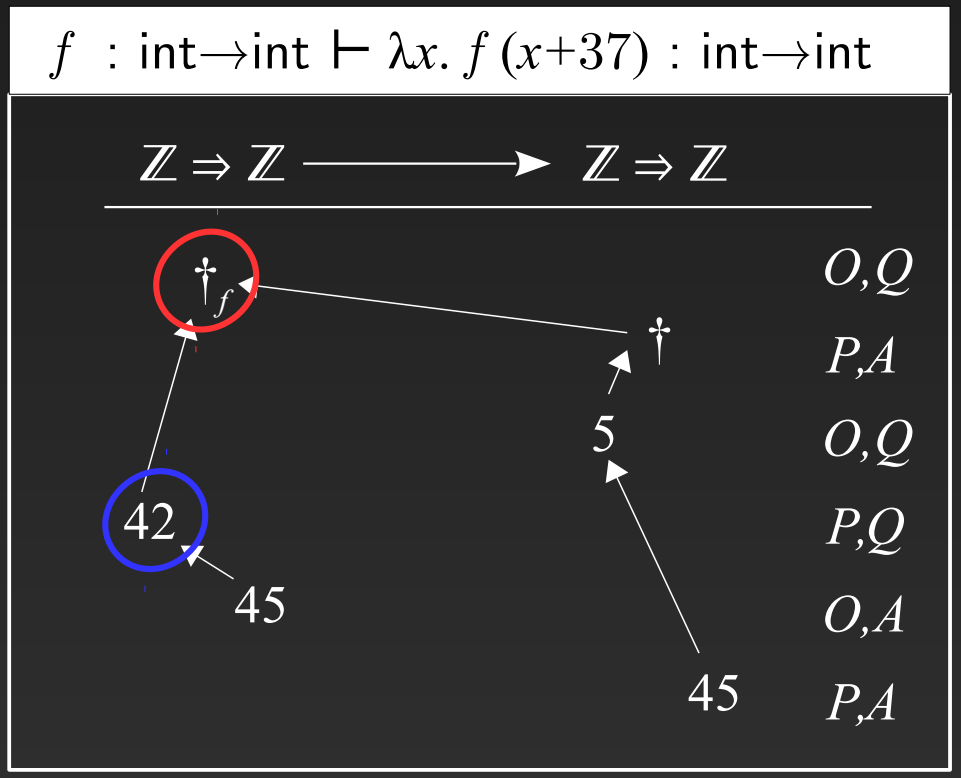


Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int

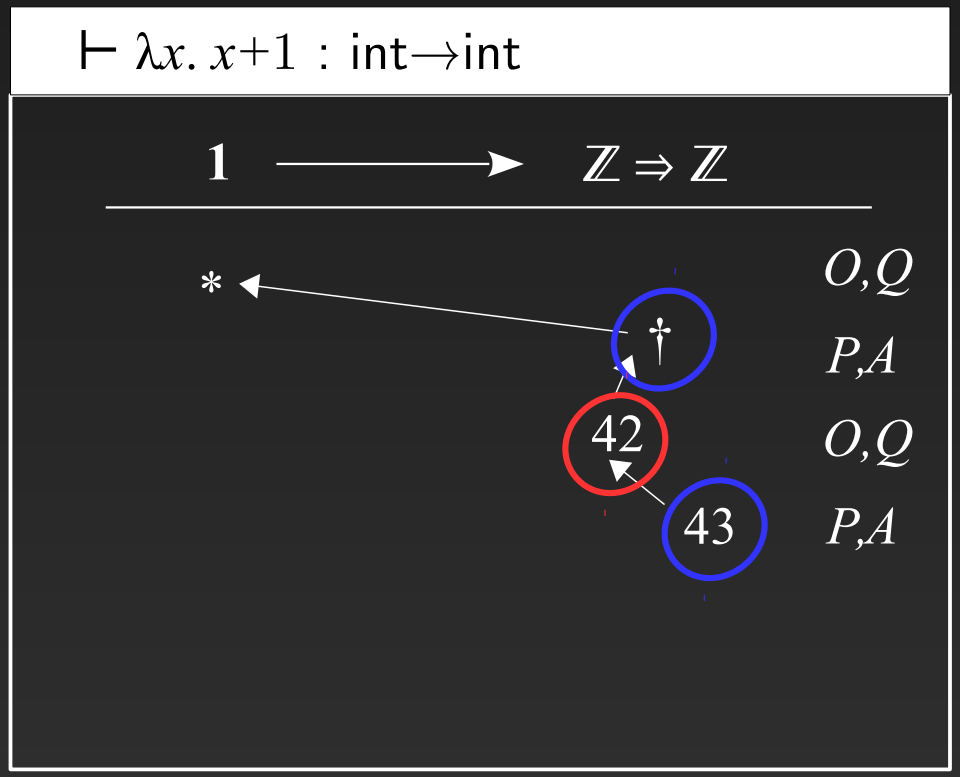


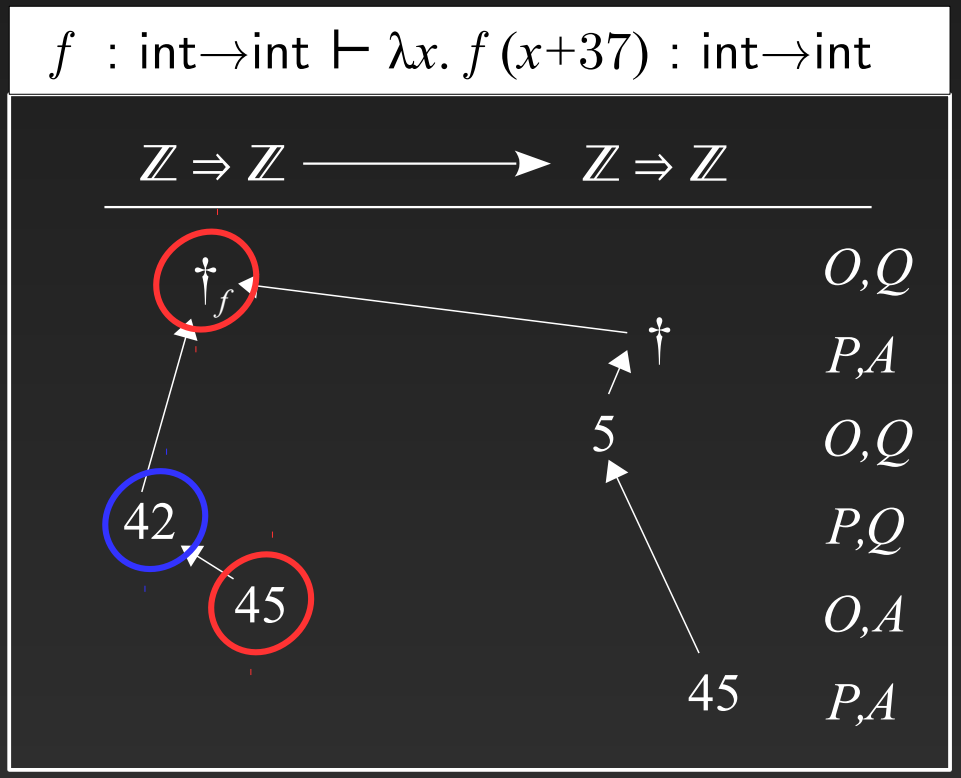


Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int

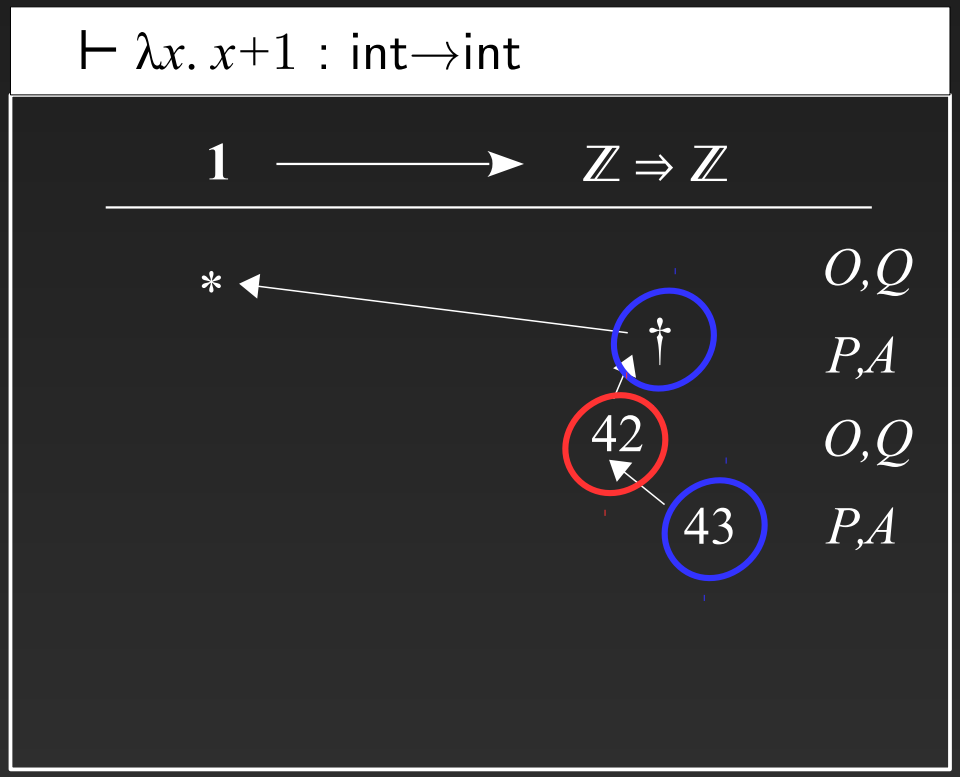


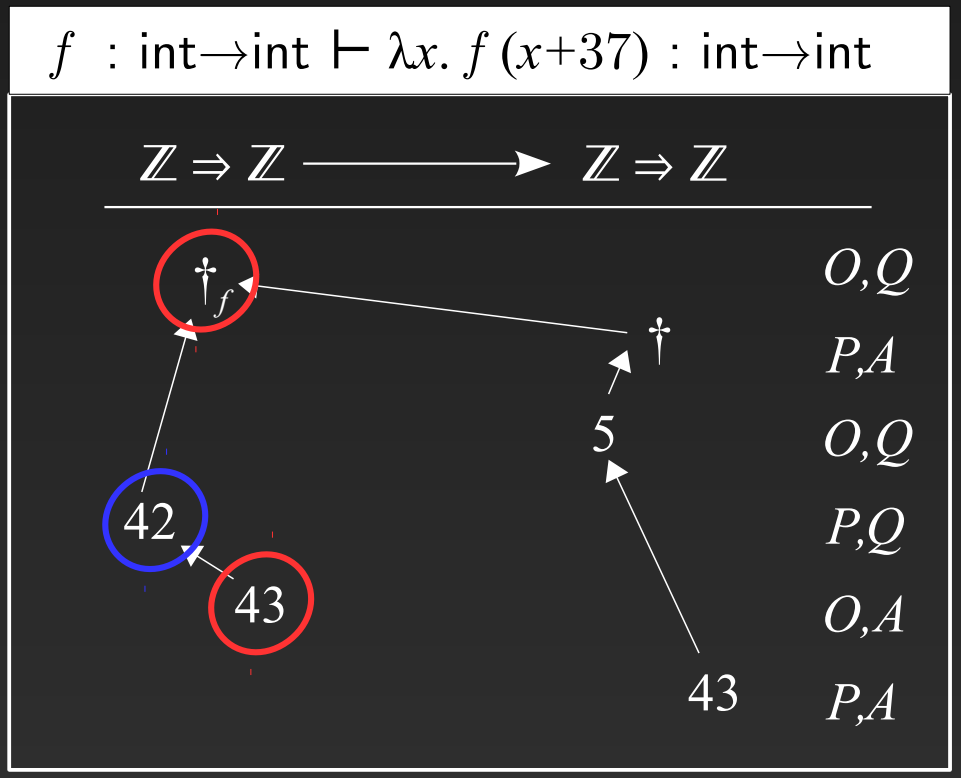


Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int

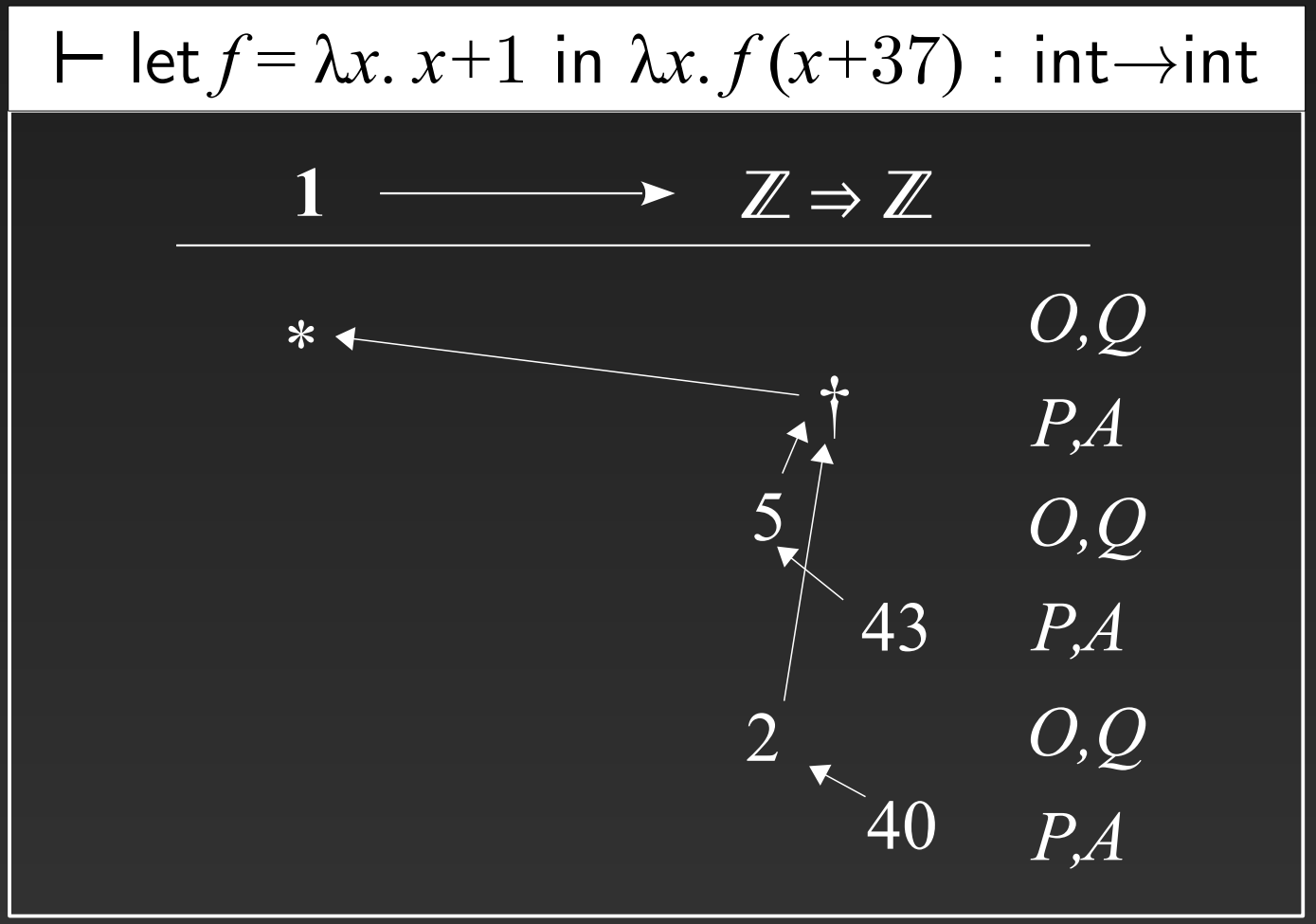




Composition

Strategies can be composed by matching O/P behaviours

→ e.g. compute let $f = \lambda x.\, x{+}1$ in $\lambda x.\, f\,(x{+}37)$: int→int



Game model construction

Game ingredients:

- move infrastructure (e.g. stores)
- sets of conditions for plays and strategies

Preparation:

- construct a category of arenas and strategy
- prove std conditions for lambda-calculus (CBV)
- model each effectful constructor (e.g. define $[\![\text{ref}]\!]$)
- prove correctness for each computational effect
- prove completeness via definability

Models for references

Games with moves that carry names and stores:

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

- pointers to moves & stores
- name privacy made explicit

[Laird '08, Murawski & T. '09, '11, '12]

Models for references

Games with moves that carry names and stores:

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

- pointers to moves & stores
- name privacy made explicit

Composition requires nominal conditions:

- privacy: one strategy cannot guess the other's names
- store access: strategies can only access the parts of the store they know

[Laird '08, Murawski & T. '09, '11, '12]

More effects: exceptions

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \qquad \frac{}{\Gamma \vdash i : \mathsf{int}} \qquad \frac{}{\Gamma \vdash \mathsf{if}_\vartheta : \mathsf{int} \to \vartheta \to \vartheta \to \vartheta} \qquad \frac{a \in \mathsf{Loc}_\vartheta}{\Gamma \vdash a : \mathsf{ref}\ \vartheta}$$

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

$$\frac{\Gamma \vdash M, N : \mathsf{ref}\ \vartheta}{\Gamma \vdash M = N : \mathsf{int}}$$

$$\vartheta, \vartheta' ::= \mathsf{unit} \mid \mathsf{int} \mid \vartheta \to \vartheta' \mid \mathsf{ref}\ \vartheta$$

$$\frac{\Gamma \vdash M : \vartheta}{\Gamma \vdash \mathsf{ref}\ M : \mathsf{ref}\ \vartheta} \qquad \frac{\Gamma \vdash M : \mathsf{ref}\ \vartheta}{\Gamma \vdash !M : \vartheta} \qquad \frac{\Gamma \vdash M : \mathsf{ref}\ \vartheta \quad \Gamma \vdash N : \vartheta}{\Gamma \vdash M := N : \mathsf{unit}}$$

More effects: exceptions

(exception names)

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \qquad \frac{}{\Gamma \vdash i : \mathsf{int}} \qquad \frac{}{\Gamma \vdash \mathsf{if}_\vartheta : \mathsf{int} \to \vartheta \to \vartheta \to \vartheta} \qquad \frac{e \in \mathsf{Exn}}{\Gamma \vdash e : \mathsf{exn}}$$

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

$$\frac{\Gamma \vdash M, N : \mathsf{exn}}{\Gamma \vdash M = N : \mathsf{int}}$$

$$\vartheta, \vartheta' ::= \mathsf{unit} \mid \mathsf{int} \mid \vartheta \to \vartheta' \mid \mathsf{exn}$$

$$v ::= () \mid i \mid \mathsf{if} \mid e \mid \lambda x.M$$

$$\frac{}{\Gamma \vdash \mathsf{exn}() : \mathsf{exn}} \qquad \frac{\Gamma \vdash M : \mathsf{exn}}{\Gamma \vdash \mathsf{raise}\, M : \vartheta} \qquad \frac{\Gamma \vdash M : \vartheta \quad \Gamma, x : \mathsf{exn} \vdash N : \vartheta}{\Gamma \vdash M \ \mathsf{handle}\ x \Rightarrow N : \vartheta}$$

Operational semantics excerpt

$$M,\ S \quad \to \quad M',\ S'$$

S : a set of exceptions

$$(\lambda x.M)v,\ S \ \to \ M[v/x],\ S$$

$$e = e',\ S \ \to \ 0/1,\ S$$

$e, e' \in$ Exn

$$\mathsf{exn}(),\ S \ \to \ e,\ S \uplus \{e\}$$

$$(\lambda x.M)(\mathsf{raise}\ e),\ S \ \to \ \mathsf{raise}\ e,\ S$$

$$v\ \mathsf{handle}\ x \Rightarrow M,\ S \quad \to \quad M[v/x],\ S$$

$$(\mathsf{raise}\ e)\ \mathsf{handle}\ x \Rightarrow M,\ S \quad \to \quad M[e/x],\ S$$

Operational semantics excerpt

$$M,\ S \quad \to \quad M',\ S'$$

S : a set of exceptions

$$(\lambda x.M)v,\ S \ \to \ M[v/x],\ S$$

$$e = e',\ S \ \to \ 0/1,\ S$$

$e, e' \in \mathsf{Exn}$

$$\mathsf{exn}(),\ S \ \to \ e,\ S \uplus \{e\}$$

$$(\lambda x.M)(\mathsf{raise}\ e),\ S \ \to \ \mathsf{raise}\ e,\ S$$

$$v\ \mathsf{handle}\ x \Rightarrow M,\ S \quad \to \quad M[v/x],\ S$$

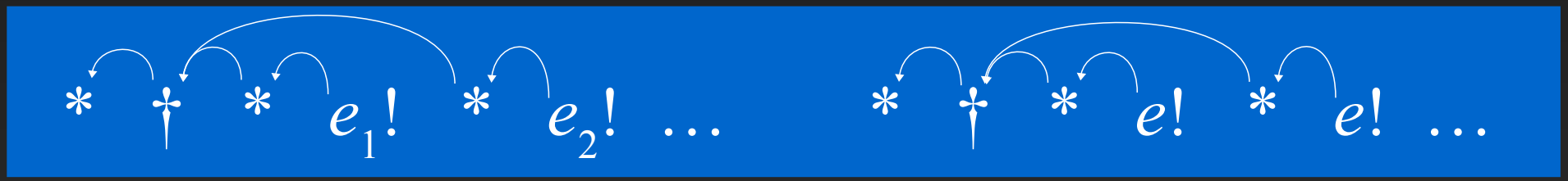
$$(\mathsf{raise}\ e)\ \mathsf{handle}\ x \Rightarrow M,\ S \quad \to \quad M[e/x],\ S$$

$$\mathsf{let}\ x = \mathsf{exn}()\ \mathsf{in}\ \lambda z.\ \mathsf{raise}\ x \ \not\equiv_{\mathsf{unit}\to\mathsf{unit}} \ \lambda z.\ \mathsf{raise}\ (\mathsf{exn}())$$

Model for exceptions

Cater for exceptional answer moves: $e!$

- can answer any open question
- model otherwise unchanged

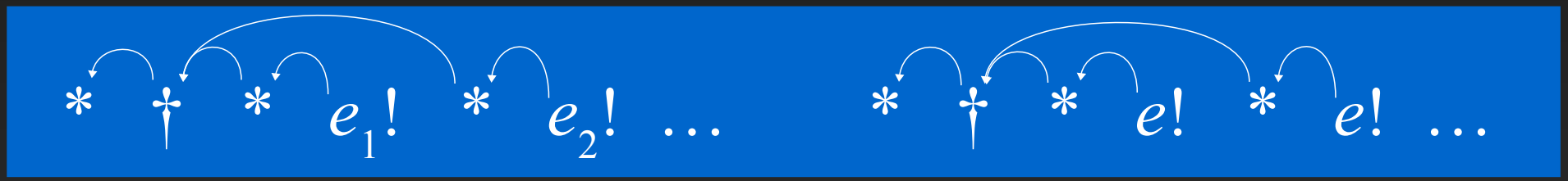


[Murawski & T. '14]

Model for exceptions

Cater for exceptional answer moves: $e!$

- can answer any open question
- model otherwise unchanged



Private exceptions:

$$\text{let } x = \text{exn}() \text{ in } \lambda z.\ \text{raise } x \quad \cong \quad \lambda z.\ \text{raise } (\text{exn}())$$

→ restrict by fresh-exception propagation

[Murawski & T. '14]

Polymorphism

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \qquad \frac{}{\Gamma \vdash i : \mathsf{int}} \qquad \frac{}{\Gamma \vdash \mathsf{if}_\vartheta : \mathsf{int} \to \vartheta \to \vartheta \to \vartheta} \qquad \frac{e \in \mathsf{Exn}}{\Gamma \vdash e : \mathsf{exn}}$$

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

$$\vartheta, \vartheta' ::= \mathsf{unit} \mid \mathsf{int} \mid \mathsf{exn} \mid \vartheta \to \vartheta'$$

$$\frac{\Gamma \vdash M, N : \mathsf{exn}}{\Gamma \vdash M = N : \mathsf{int}}$$

$$v ::= () \mid i \mid \mathsf{if} \mid e \mid \lambda x.M$$

$$\frac{}{\Gamma \vdash \mathsf{exn}() : \mathsf{exn}} \qquad \frac{\Gamma \vdash M : \mathsf{exn}}{\Gamma \vdash \mathsf{raise}\, M : \vartheta} \qquad \frac{\Gamma \vdash M : \vartheta \qquad \Gamma, x : \mathsf{exn} \vdash N : \vartheta}{\Gamma \vdash M \ \mathsf{handle}\ x \Rightarrow N : \vartheta}$$

Polymorphism

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \qquad \frac{}{\Gamma \vdash i : \mathsf{int}} \qquad \frac{}{\Gamma \vdash \mathsf{if}_\vartheta : \mathsf{int} \to \vartheta \to \vartheta \to \vartheta}$$

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

$$\vartheta, \vartheta' ::= \mathsf{unit} \mid \mathsf{int} \mid \alpha \mid \vartheta \to \vartheta' \mid \forall \alpha.\, \vartheta$$

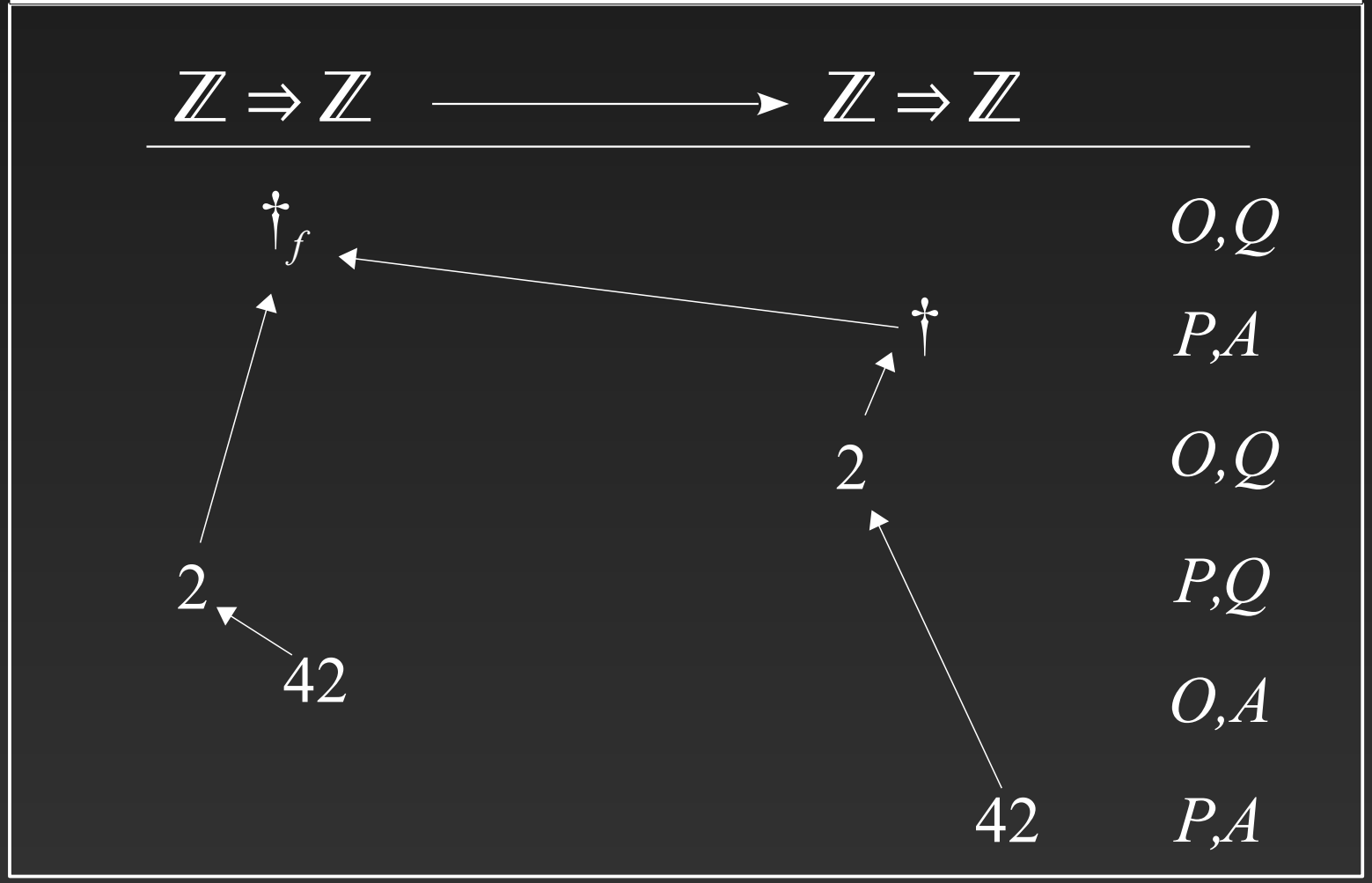
$$\frac{\Gamma \vdash M : \vartheta}{\Gamma \vdash \Lambda \alpha.\, M : \forall \alpha.\, \vartheta} \qquad \frac{\Gamma \vdash M : \forall \alpha.\, \vartheta}{\Gamma \vdash M\vartheta' : \vartheta[\vartheta'/\alpha]}$$

$$v ::= () \mid i \mid \mathsf{if} \mid \Lambda \alpha.\, M \mid \lambda x.M$$

Model for polymorphism

Names for types and polymorphic values

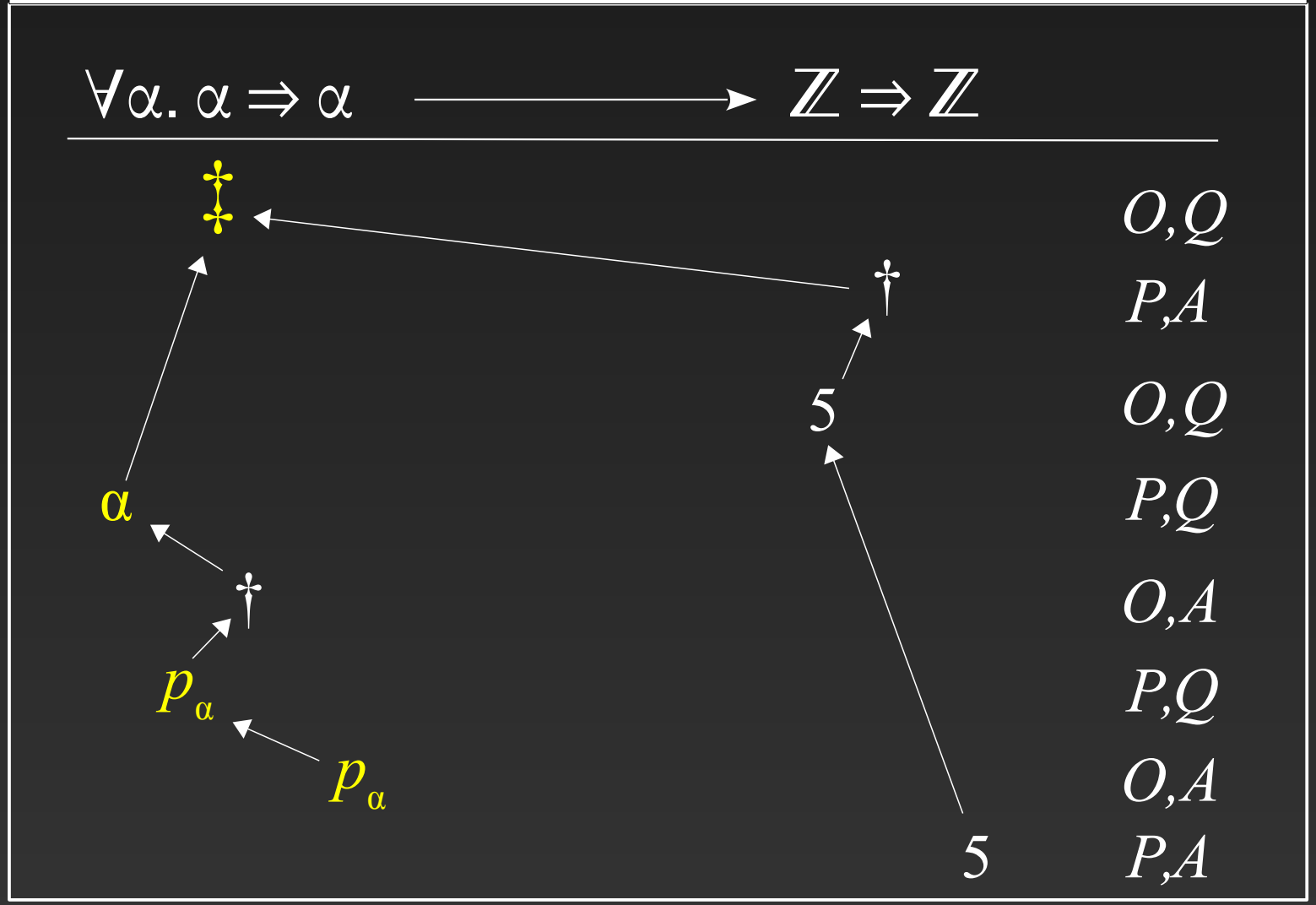
$f : \mathsf{int} \to \mathsf{int} \vdash \lambda x.\, fx : \mathsf{int} \to \mathsf{int}$



Model for polymorphism

Names for types and polymorphic values

$$f : \forall\alpha.\alpha \to \alpha \vdash \lambda x.\, f(\text{int})(x) : \text{int} \to \text{int}$$

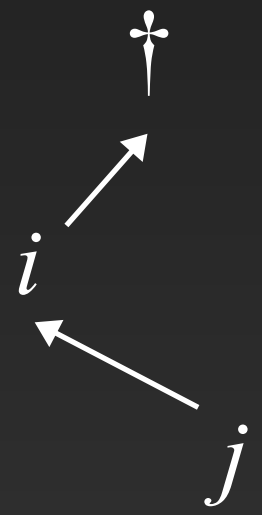


Model for polymorphism

We use names to abstract away types and values

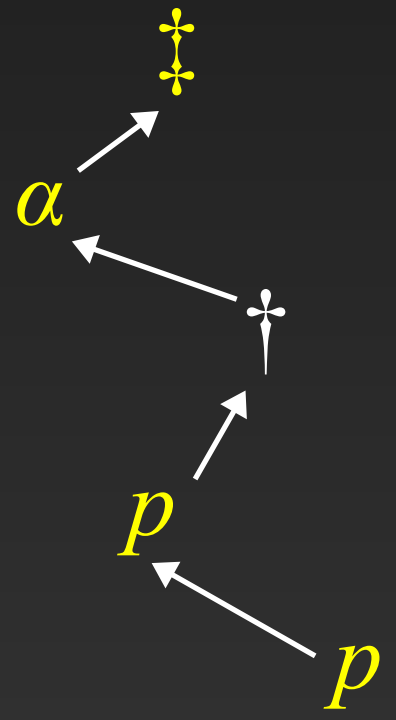
- impose uniform polymorphic behaviour
- not as clean if we also have references (*type disclosure*)

$[\![\mathsf{int} \to \mathsf{int}]\!] =$



(all $i,j \in \mathbb{Z}$)

$[\![\forall \alpha.\ \alpha \to \alpha]\!] =$



Names for:

- type variables $\alpha \in \mathcal{N}_{\mathrm{TVar}}$
- polymorphic values $p \in \mathcal{N}_{\alpha}$

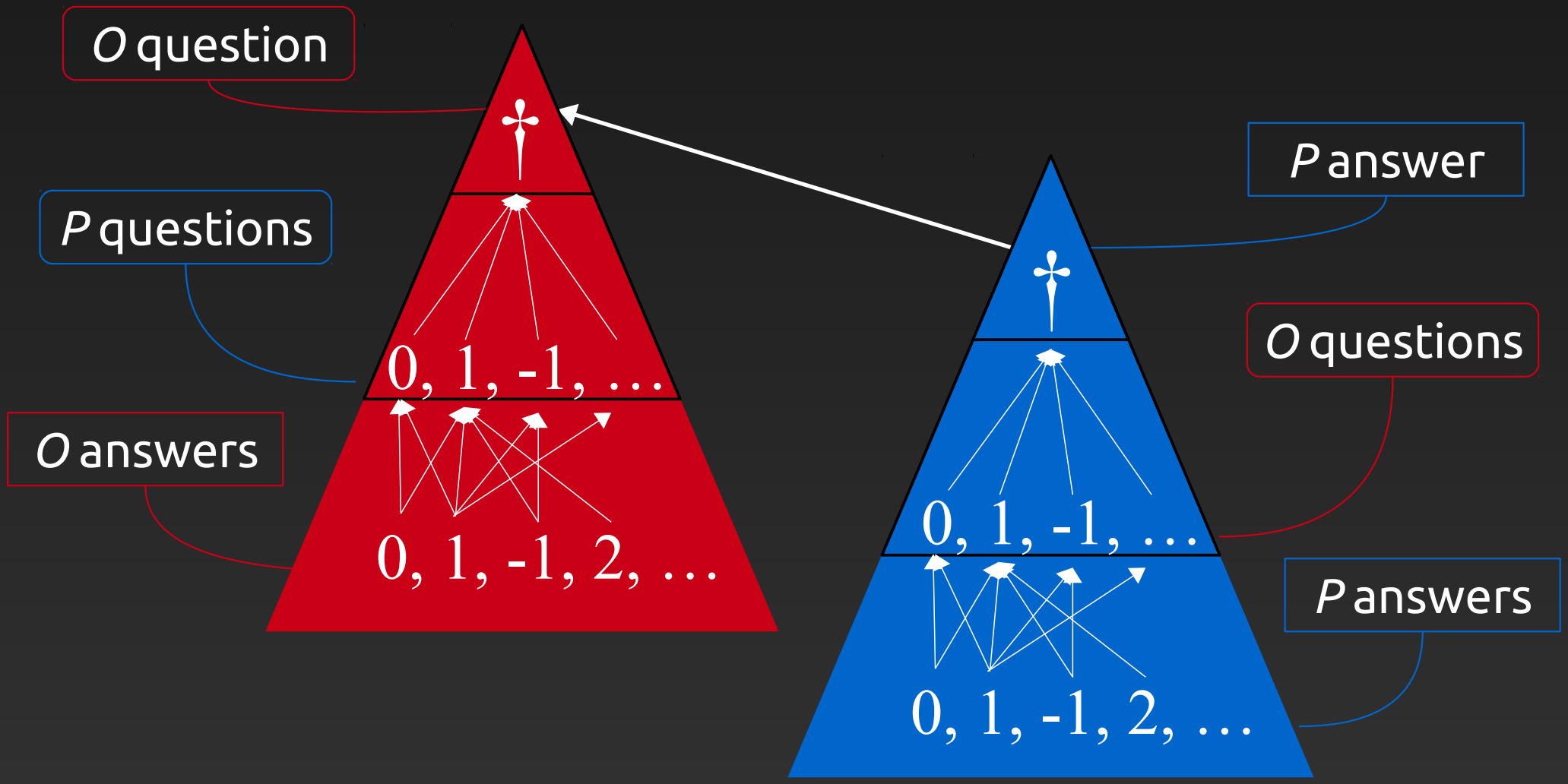
[Laird '10; Jaber & T. '16]

Games and traces

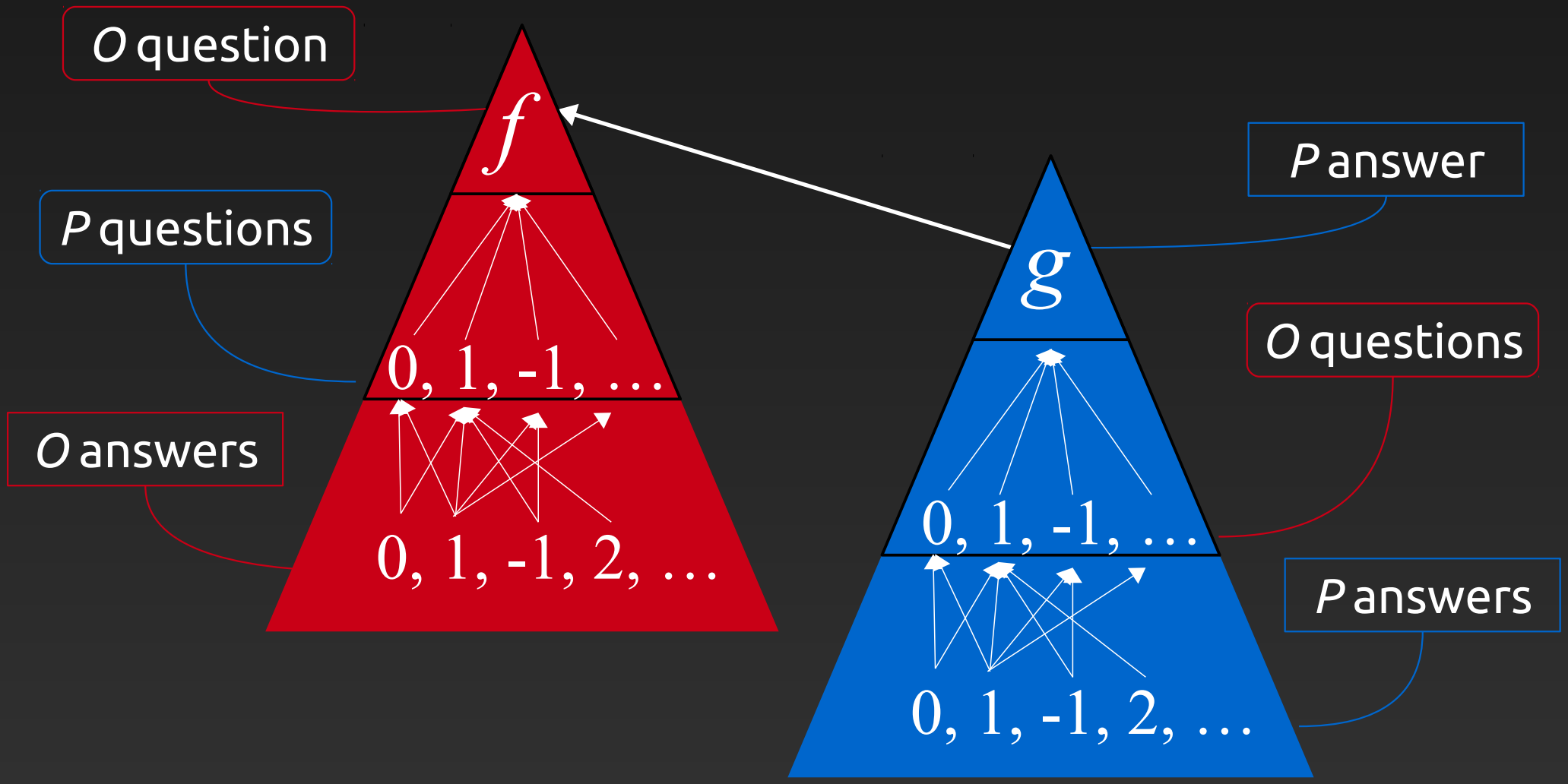
Games can be given a simpler, operational presentation:

- pointer structure → named functions
- definition: denotational/compositional
  → operational/executable

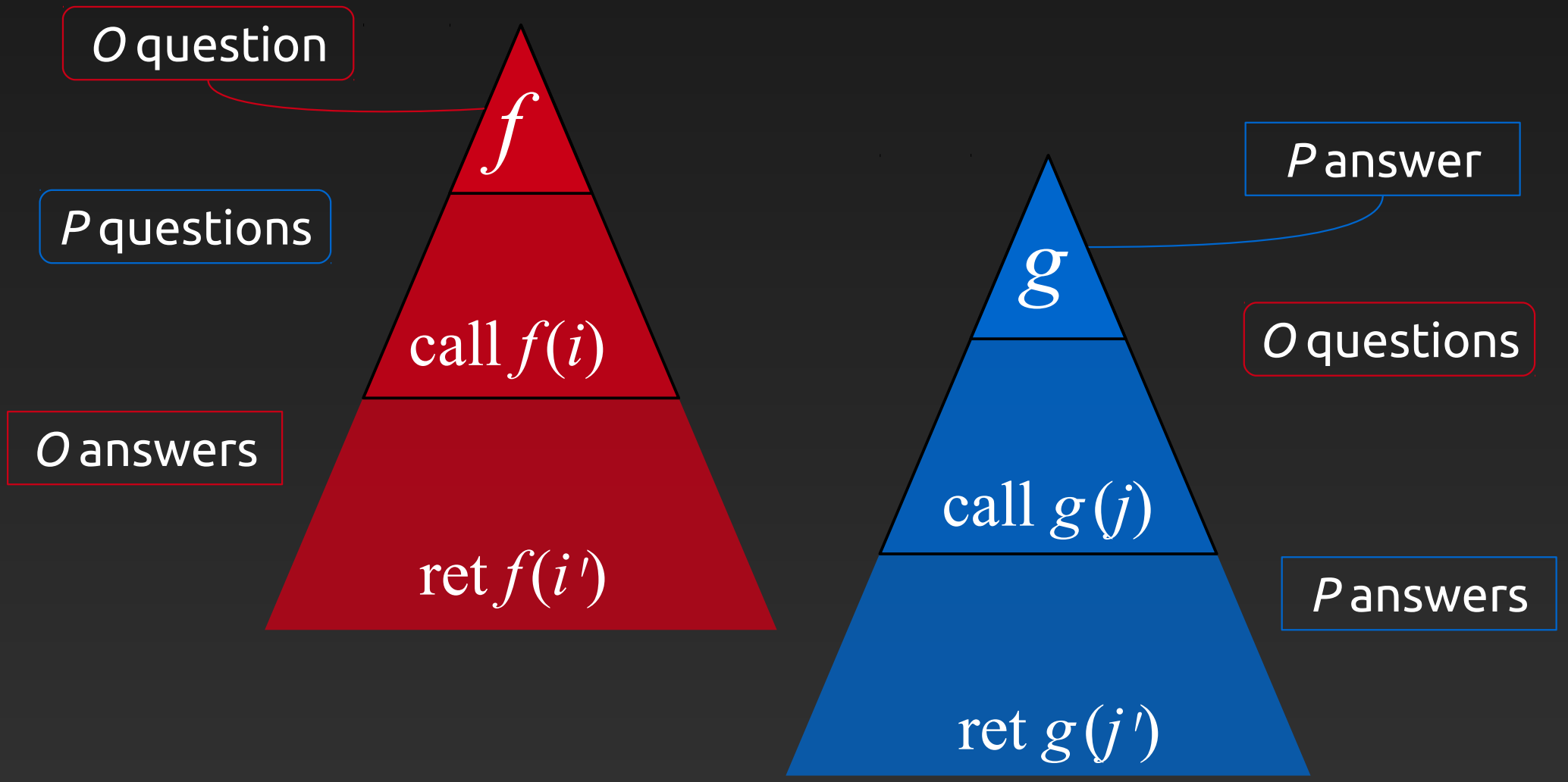
$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$



$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$



$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$



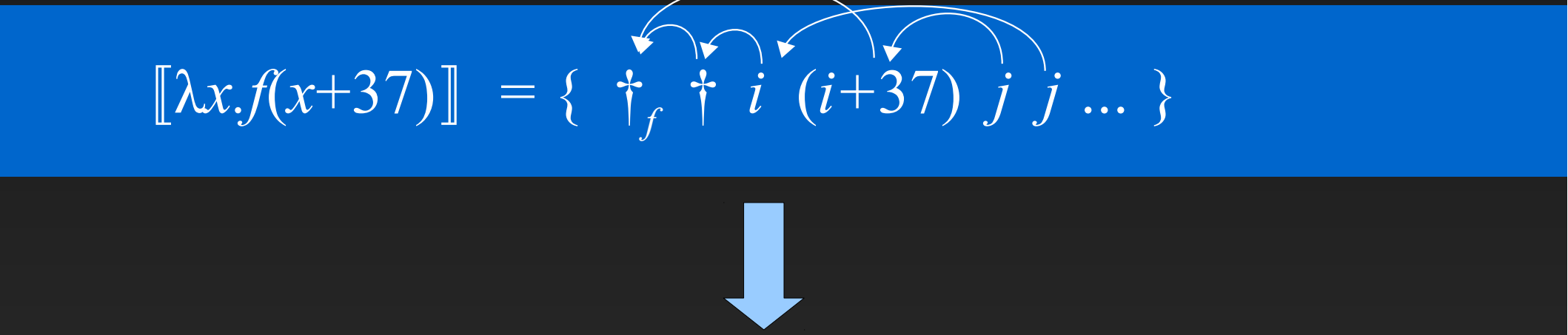
$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$

$$f \;\; \mathrm{call}(f,8) \;\; \mathrm{ret}(f,i) \;\; g \;\; \mathrm{call}(g,j) \;\; \mathrm{ret}(g,i)$$

$$(\;\text{let}\;\; x = f\,8 \;\;\text{in}\;\; \lambda z.x\;)$$

Example revisited

$$f : \mathsf{int} \to \mathsf{int} \vdash \lambda x.\, f(x{+}37) : \mathsf{int} \to \mathsf{int}$$



$$f\ g\ \mathrm{call}\, g(i)\ \mathrm{call}\, f(i{+}37)\ \mathrm{ret}\, f(j)\ \mathrm{ret}\, g(j)\ \ldots$$

Operational games

Full Abstraction Factory

Alan Jeffrey | Julian Rathke

Introduction

Papers

Papers from the Full Abstraction Factory

1. **Full Abstraction for Polymorphic Pi-Calculus.** A. S. A. Jeffrey and J. Rathke. In *Theoretical Computer Science*. 2007. To appear. Available on-line. Extended abstract in *Proc. Foundations of Software Science and Computation Structures*. Springer-Verlag. 2005. pp. 266-281. Available on-line.
2. **A Fully Abstract May Testing Semantics for Concurrent Objects.** A. S. A. Jeffrey and J. Rathke. In *Theoretical Computer Science*. vol. 338. 2005. pp. 17-63. Available on-line.
3. **Contextual Equivalence for Higher-Order Pi-Calculus Revisited.** A. S. A. Jeffrey and J. Rathke. In *Logical Methods in Computer Science*. 1 (1:4). 2005. pp. 1-22. Available on-line. Extended abstract in *Proc. Mathematical Foundations of Programming Semantics*. Elsevier. 2003. Available on-line.
4. **Java Jr.: Fully Abstract Trace Semantics for a Core Java Language.** A. S. A. Jeffrey and J. Rathke. In *Proc. European Symposium on Programming*. Springer-Verlag. 2005. pp. 423-438. Available on-line.
5. **A Theory of Bisimulation for a Fragment of Concurrent ML with Local Names.** A. S. A. Jeffrey and J. Rathke. In *Theoretical Computer Science*. vol. 323. 2004. pp. 1-48. Available on-line. Extended abstract in *Proc. IEEE Logic in Computer Science*. IEEE Press. 2000. pp. 311-321. Available on-line.
6. **Towards a Theory of Bisimulation for Local Names.** A. S. A. Jeffrey and J. Rathke. In *Proc. IEEE Logic in Computer Science*. IEEE Press. 1999. pp. 56--66. Available on-line.
7. **A Fully Abstract Semantics for a Nondeterministic Functional Language with Monadic Types.** A. S. A. Jeffrey. In *Theoretical Computer Science*. vol. 228. 1999. pp. 105-150. Available on-line. Extended abstract in *Proc. Mathematical Foundations of Programming Semantics*. Elsevier. 1995. Available on-line.
8. **Semantics for Core Concurrent ML Using Computation Types.** A. S. A. Jeffrey. In *Proc. Higher Order Operational Techniques in Semantics*. Cambridge University Press. 1997. pp. 55-89. Available on-line.
9. **A Fully Abstract Semantics for a Concurrent Functional Language with Monadic Types.** A. S. A. Jeffrey. In *Proc. IEEE Logic in Computer Science*. IEEE Press. 1995. pp. 255--264. Available on-line.
10. **A Fully Abstract Semantics for Concurrent Graph Reduction.** A. S. A. Jeffrey. In *Proc. IEEE Logic in Computer Science*. IEEE Press. 1994. pp. 82-91. Available on-line.

This material is partly based upon work supported by the National Science Foundation under Grant No. 0430175, by the Engineering and Physical Sciences Research Council grants *Foundations for the Integration of Concurrent, Distributed and Functional Computation* and *Linear Type Systems for Concurrent Systems*, and by the Nuffield Foundation grant *A Semantic Study of Behavioural Properties for Systems of Distributed Objects*.

[Jaffrey & Rathke '04; Laird '07; Ghica & T. '12; Levy & Staton '14; Jaber '15]

Operational games

A marriage of games and traces whereby:

- pointers are replaced by names:
  * plays are (bracketed) sequences of name-calls and -returns
- the arena structure is removed altogether:
  * arenas become *flat*: integers, units and names (functions, references, exceptions, etc.)
  * composition becomes more involved, as move/name ownership is now *dynamic*

[Jaffrey & Rathke '04; Laird '07; Ghica & T. '12; Levy & Staton '14; Jaber '15]

Operational games

Defined operational, via an open transition system

- Configurations

$\langle\ M, S, C, \mathcal{U}\ \rangle$ (for P) $\qquad$ $\langle\ S, C, \mathcal{U}\ \rangle$ (for O)

- Transition rules

[Int] if $M, S \to M', S'$ then $\langle M, S, C, \mathcal{U}\rangle \longrightarrow \langle M', S', C, \mathcal{U}\rangle$

[OQ] $\langle S, C, \mathcal{U}\rangle \xrightarrow{\text{call } f(v), S'} \langle \mathcal{U}(f),\ S[S'], f::C, \mathcal{U}'\rangle$

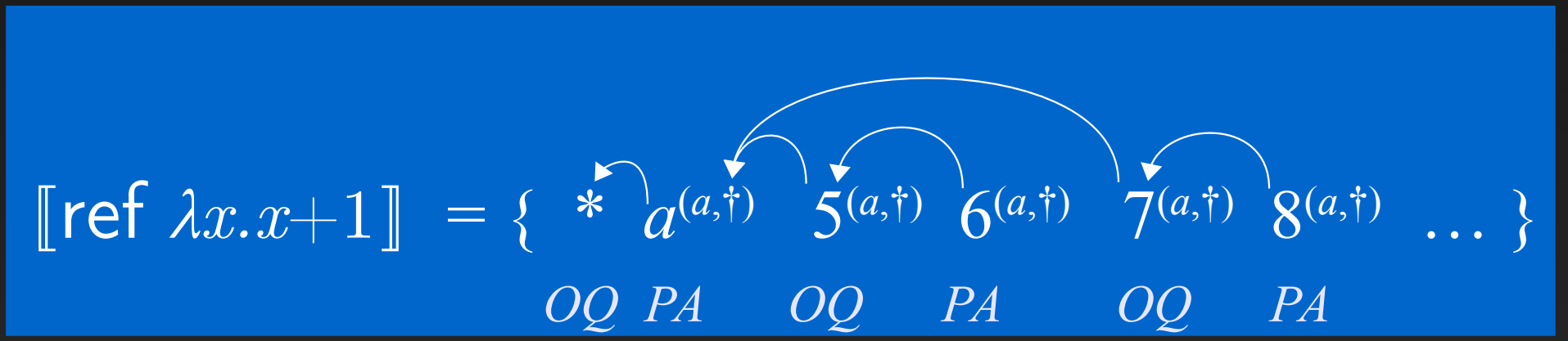
[PA] $\langle v, S, f::C, \mathcal{U}\rangle \xrightarrow{\text{ret } f(v'), S'} \langle\ S, C, \mathcal{U}'\rangle$

[PQ] $\langle E[f\,v], S, C, \mathcal{U}\rangle \xrightarrow{\text{call } f(v'), S'} \langle\ S, (f,E)::C, \mathcal{U}'\rangle$

[OA] $\langle S, (f,E)::C, \mathcal{U}\rangle \xrightarrow{\text{ret } f(v), S'} \langle E[v], S[S'], C, \mathcal{U}'\rangle$

Example revisited

$\vdash \mathsf{ref}\ (\lambda x^{\mathsf{int}}.x{+}1) : \mathsf{ref}\,(\mathsf{int}{\to}\mathsf{int})$



$* \quad a^{(a,f)} \quad \mathrm{call}\,(f,5)^{(a,f')} \quad \mathrm{ret}\,(f,6)^{(a,f'')} \quad \mathrm{call}\,(f,7)^{(a,f''')} \quad \mathrm{ret}\,(f,8)^{(a,f'''')}$

Recap and further work

Nominal game semantics for modelling HO programs

- fragments of ML and Java
- also low-level languages (C-like)

Denotational and operational presentations

- “simple” games, good for non-experts
- amenable to trace-based techniques

Further on:

- concurrency (e.g. names for threads)
- games and effects systems

Recap and further work

Nominal game semantics for modelling HO programs

- fragments of ML and Java
- also low-level languages (C-like)

Denotational and operational presentations

- “simple” games, good for non-experts
- amenable to trace-based techniques

Further on:

- concurrency (e.g. names for threads)
- games and effects systems

Thanks!