

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

IMS, Singapore, September 2016

Supported by a Royal Academy of Engineering Research Fellowship

what this talk is about

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

Look into semantic models capturing these effects

$$\llbracket - \rrbracket : \text{Syntax} \rightarrow \mathcal{M}$$

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

$$\begin{array}{c} \overline{\Gamma \vdash () : \text{unit}} \quad \overline{\Gamma \vdash i : \text{int}} \quad \overline{\Gamma \vdash \text{if}_{\vartheta} : \text{int} \rightarrow \vartheta \rightarrow \vartheta \rightarrow \vartheta} \\ \overline{\Gamma, x : \vartheta \vdash x : \vartheta} \quad \frac{\Gamma, x : \vartheta \vdash M : \vartheta'}{\Gamma \vdash \lambda x^{\vartheta}. M : \vartheta \rightarrow \vartheta'} \quad \frac{\Gamma \vdash M : \vartheta \rightarrow \vartheta' \quad \Gamma \vdash N : \vartheta}{\Gamma \vdash MN : \vartheta'} \end{array}$$

$$\vartheta, \vartheta' ::= \text{unit} \mid \text{int} \mid \vartheta \rightarrow \vartheta'$$

Setting: HO programs + effects

typed locations
(location names)

$$\begin{array}{c}
 \frac{}{\Gamma \vdash () : \text{unit}} \quad \frac{}{\Gamma \vdash i : \text{int}} \quad \frac{}{\Gamma \vdash \text{if}_\vartheta : \text{int} \rightarrow \vartheta \rightarrow \vartheta \rightarrow \vartheta} \quad \frac{a \in \text{Loc}_\vartheta}{\Gamma \vdash a : \text{ref } \vartheta} \\
 \\
 \frac{}{\Gamma, x : \vartheta \vdash x : \vartheta} \quad \frac{\Gamma, x : \vartheta \vdash M : \vartheta'}{\Gamma \vdash \lambda x^\vartheta. M : \vartheta \rightarrow \vartheta'} \quad \frac{\Gamma \vdash M : \vartheta \rightarrow \vartheta' \quad \Gamma \vdash N : \vartheta}{\Gamma \vdash MN : \vartheta'} \\
 \\
 \frac{\Gamma \vdash M, N : \text{ref } \vartheta}{\Gamma \vdash M = N : \text{int}}
 \end{array}$$

$$\vartheta, \vartheta' ::= \text{unit} \mid \text{int} \mid \vartheta \rightarrow \vartheta' \mid \text{ref } \vartheta$$

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

Operational semantics (examples)

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

S stores locations
+ their values

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

$$a = a', S \rightarrow 0/1, S \quad a, a' \in \text{Loc}_\theta$$

$$\text{ref } 0, S \rightarrow a, S \uplus [a \mapsto 0] \quad a \in \text{Loc}_{\text{int}}$$

$$\text{ref } (\lambda x.x+1), S \rightarrow a, S \uplus [a \mapsto \lambda x.x+1] \quad a \in \text{Loc}_{\text{int} \rightarrow \text{int}}$$

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

$$\rightarrow \dots \rightarrow [a \mapsto \lambda x.(!a)x], (!a)5 \rightarrow [a \mapsto \lambda x.(!a)x], (\lambda x.(!a)x)5 \rightarrow \dots$$

Modelling problem

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

$$\llbracket - \rrbracket : \text{Syntax} \longrightarrow \mathcal{M}$$

such that the modelling is sound and complete:

$$\llbracket P \rrbracket = \llbracket P' \rrbracket \iff P \cong P'$$

Why?

- reasoning about programs \rightarrow model checking
[e.g. model checking equivalence, see [Coneqct](#)]

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

Modelling using game semantics

- 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

free variables

program

output type

$$x_1:\theta_1, \dots, x_n:\theta_n \vdash M:\theta$$

input types

$$[[M]] : [[\theta_1, \dots, \theta_n]] \longrightarrow [[\theta]]$$

strategy

move arenas

Arenas of moves

$$[[M]] : [[T_1, \dots, T_n]] \longrightarrow [[T]]$$

strategy

move arenas

Arenas of moves

$$[[M]] : [[T_1, \dots, T_n]] \longrightarrow [[T]]$$

move arenas

moves

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

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

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

...

Arenas of moves

$$[[M]] : [[T_1, \dots, T_n]] \longrightarrow [[T]]$$



move arenas

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

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

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

\mathcal{N}_θ a set of *names*:

- infinitely many
- comparable for equality only

Example games

$\vdash 42 : \text{int}$

$1 \longrightarrow \mathbb{Z}$

*

$0, Q$

Example games

$\vdash 42 : \text{int}$

$1 \longrightarrow \mathbb{Z}$

*

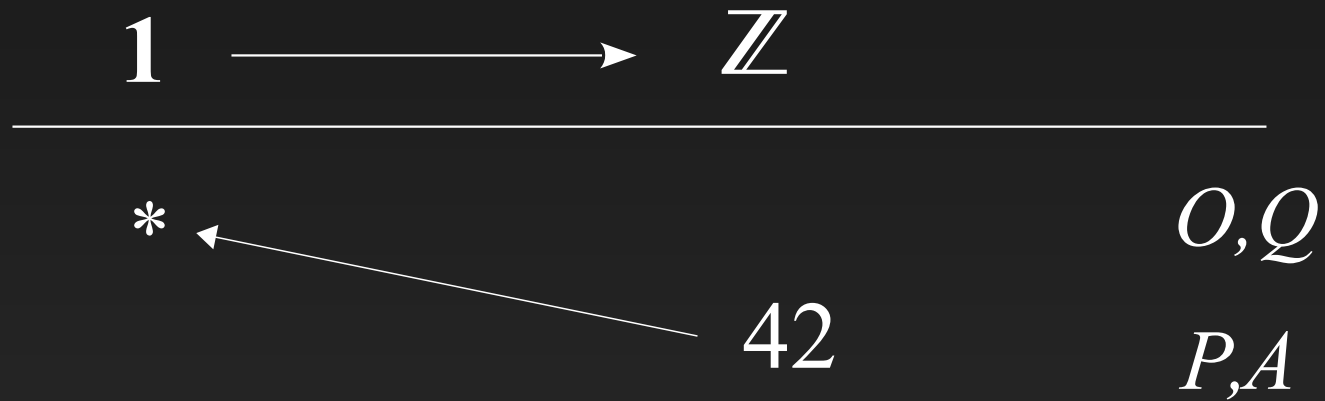
O, Q

42

P, A

Example games

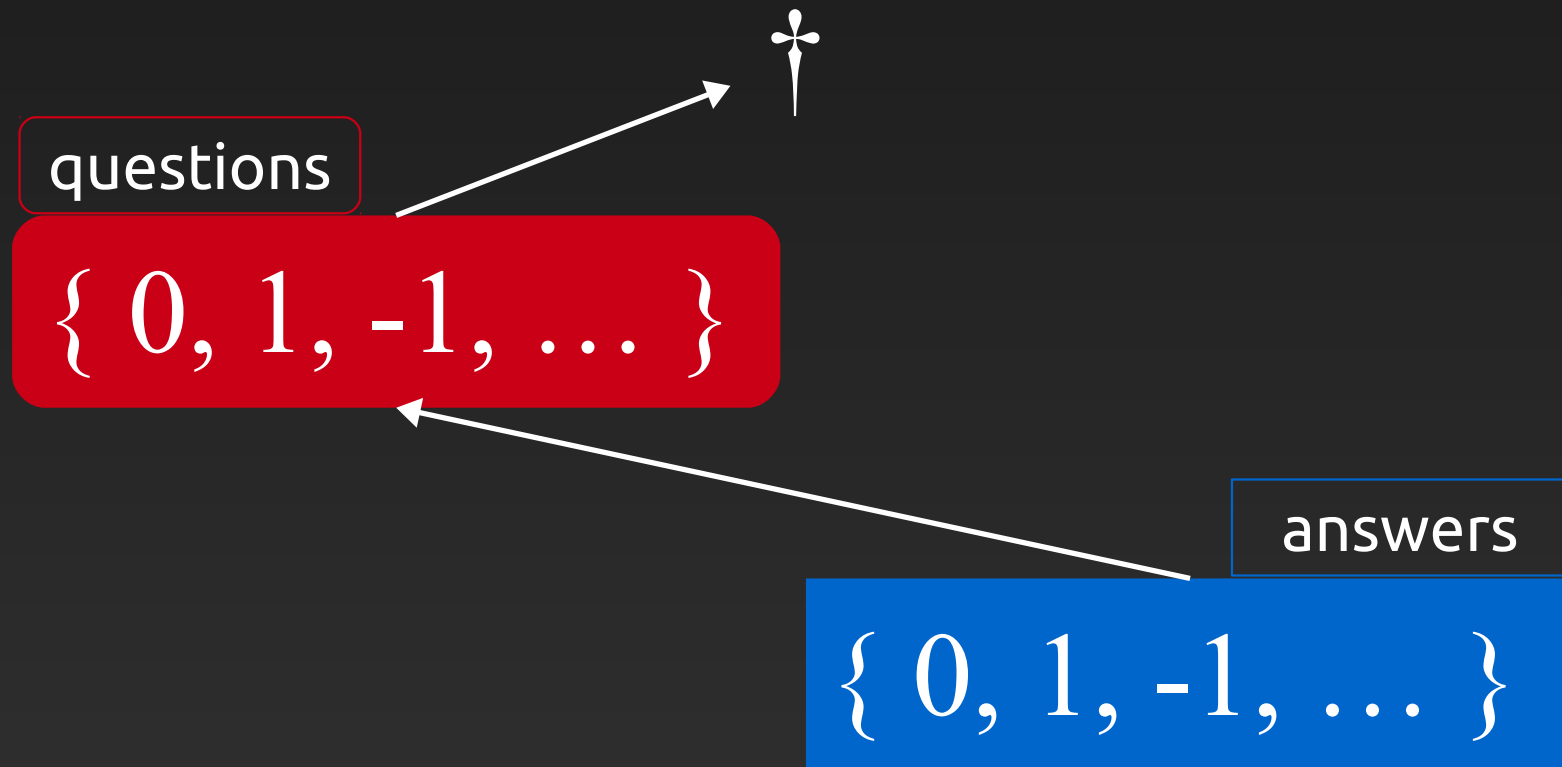
$\vdash 42 : \text{int}$



$$[[42]] = \{ \begin{array}{c} \text{*} \quad \text{42} \\ \text{OQ} \quad \text{PA} \end{array} \}$$

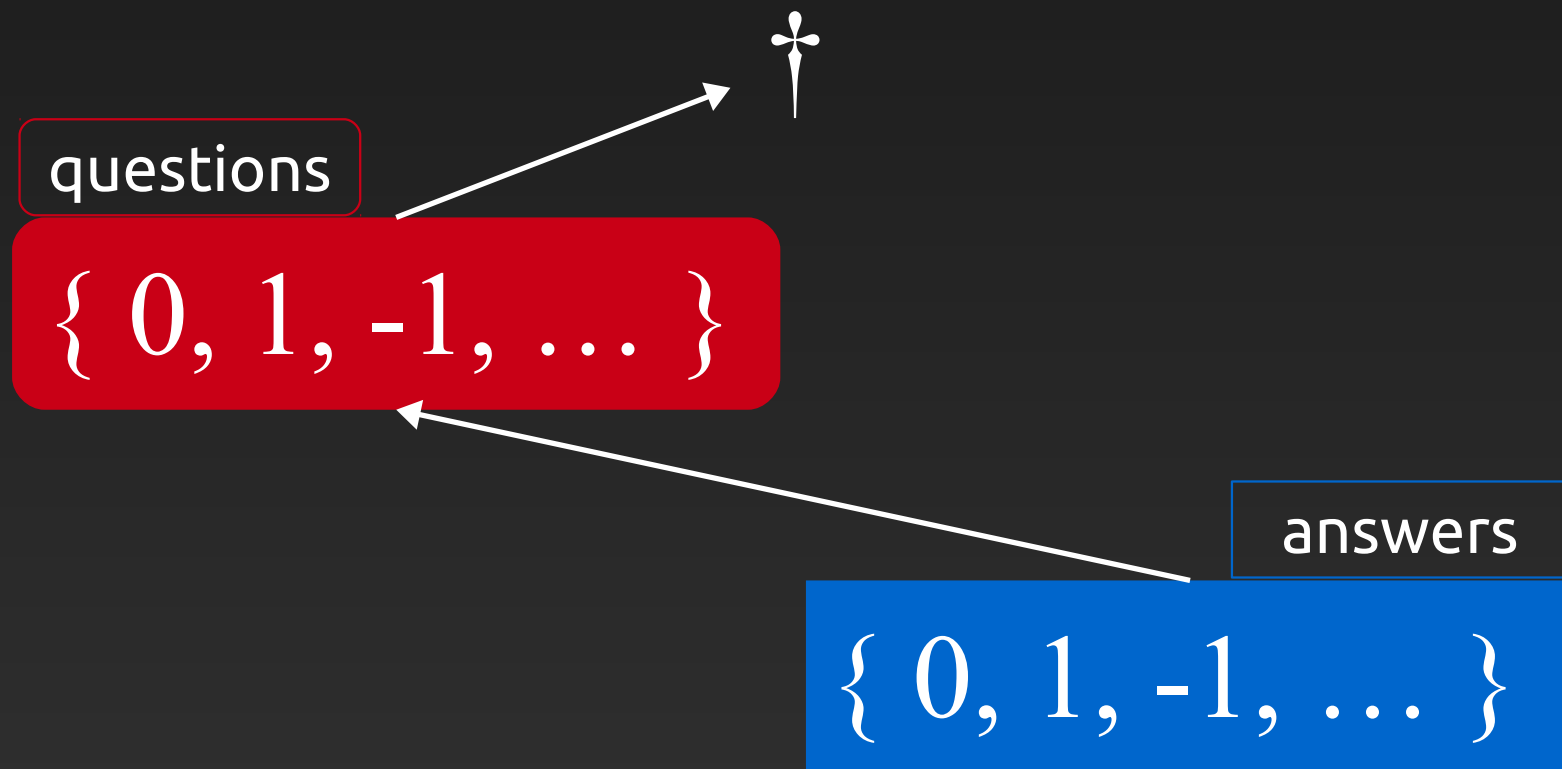
A higher-order arena

$\llbracket \text{int} \rightarrow \text{int} \rrbracket$



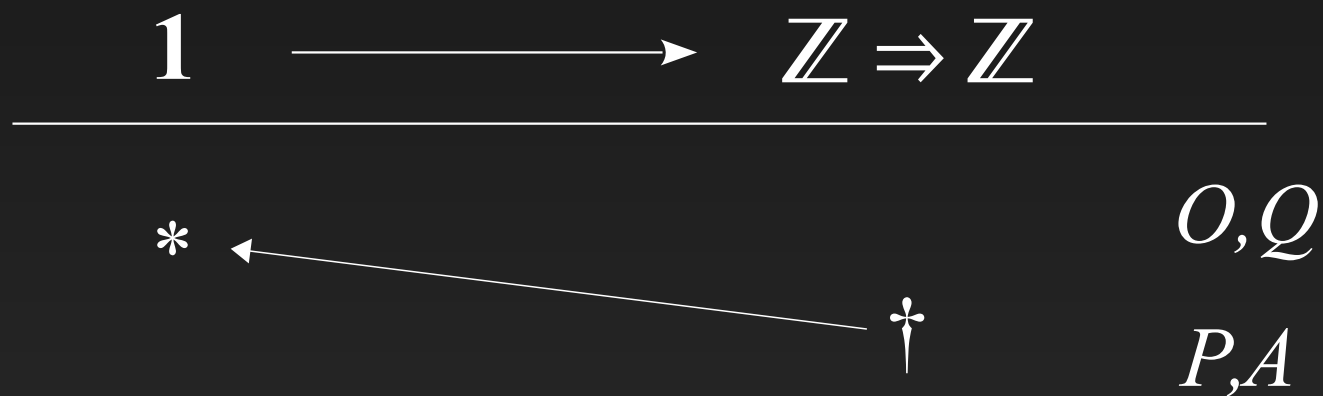
A higher-order arena

$$\llbracket \text{int} \rightarrow \text{int} \rrbracket = \mathbb{Z} \Rightarrow \mathbb{Z}$$



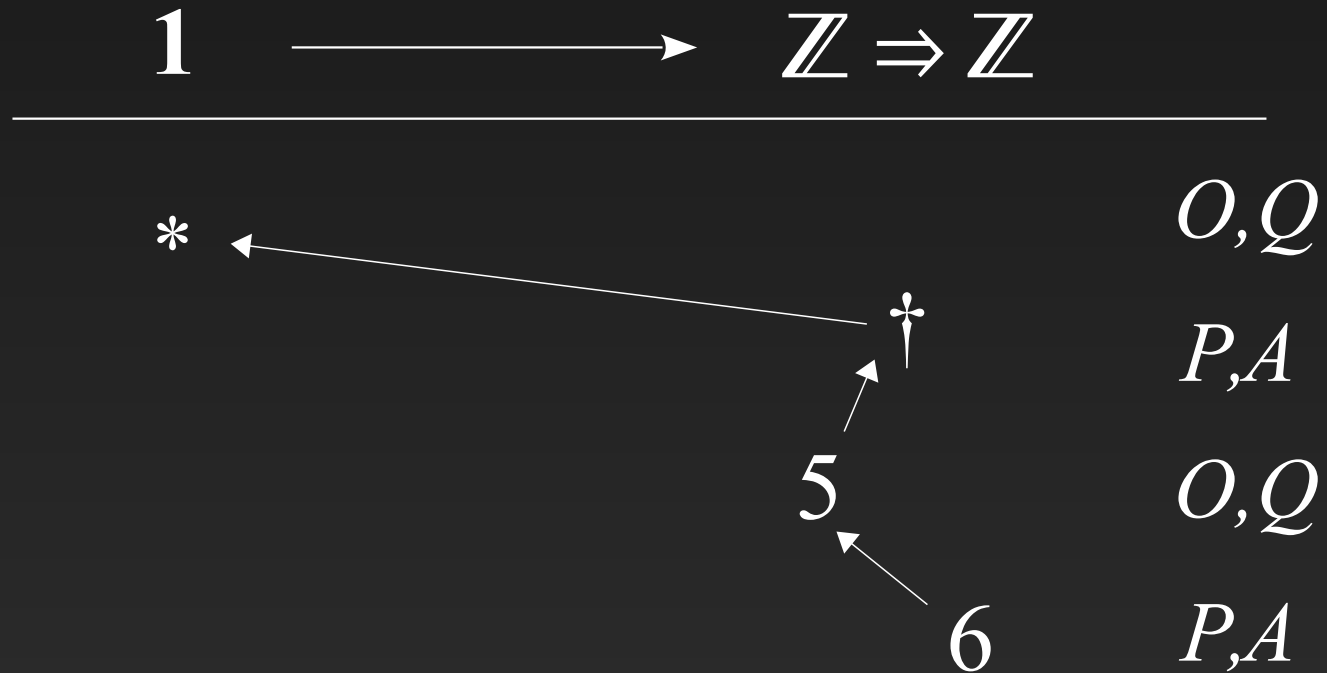
Example games

$\vdash \lambda x. x+1 : \text{int} \rightarrow \text{int}$



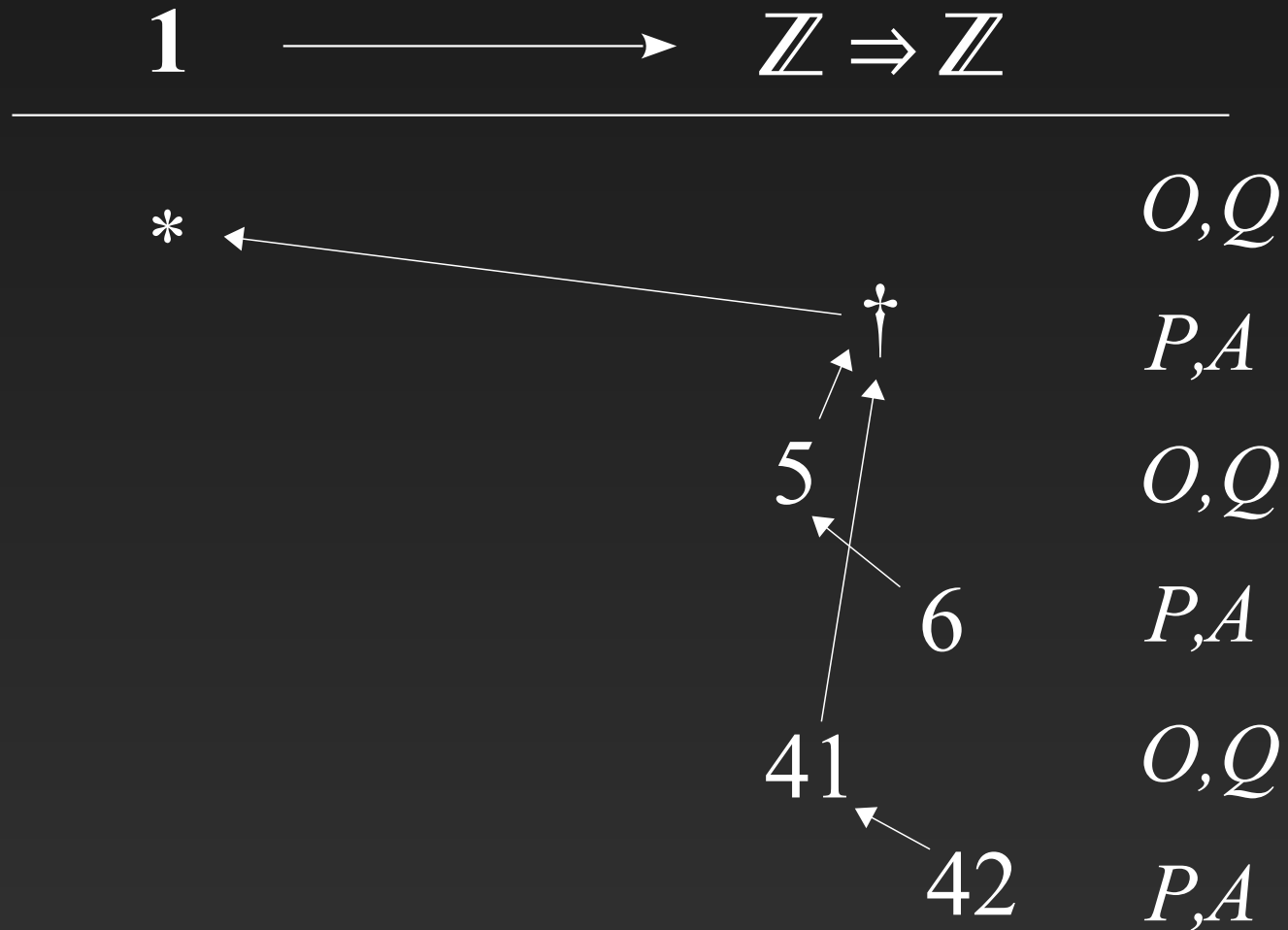
Example games

$\vdash \lambda x. x+1 : \text{int} \rightarrow \text{int}$



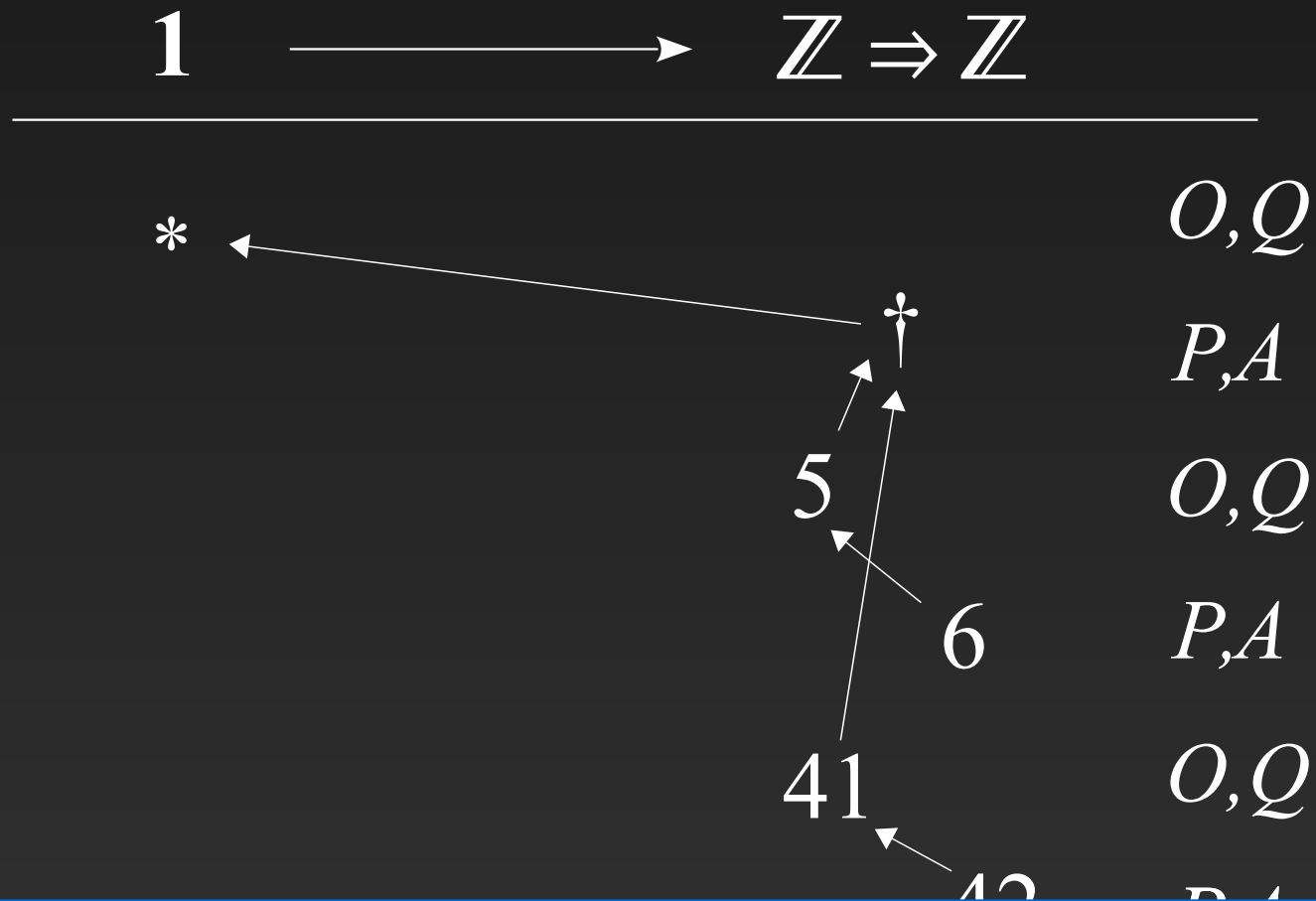
Example games

$\vdash \lambda x. x+1 : \text{int} \rightarrow \text{int}$



Example games

$\vdash \lambda x. x+1 : \text{int} \rightarrow \text{int}$



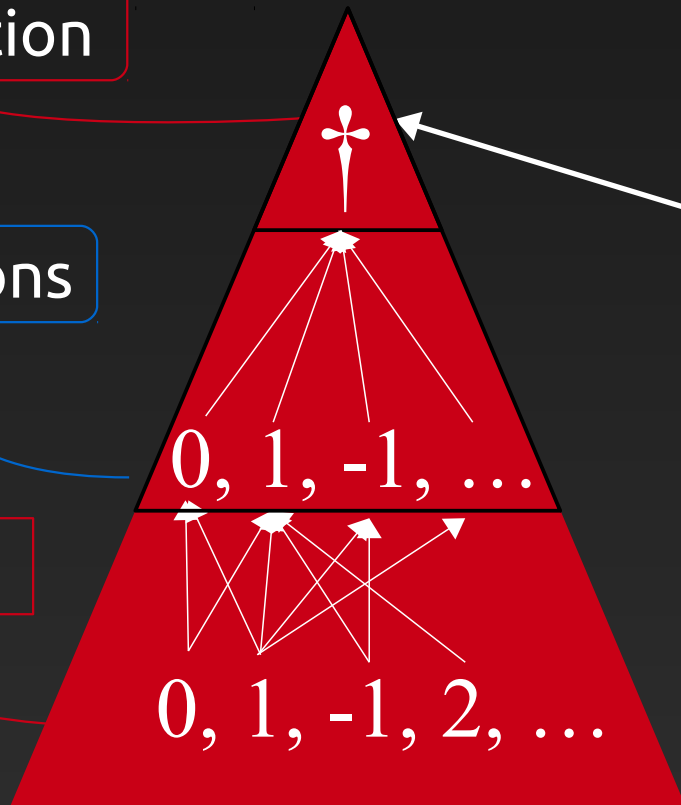
$$\llbracket \lambda x. x+1 \rrbracket = \{ * \ \dagger \ i \ i+1 \ \dots \}$$

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

O question

P questions

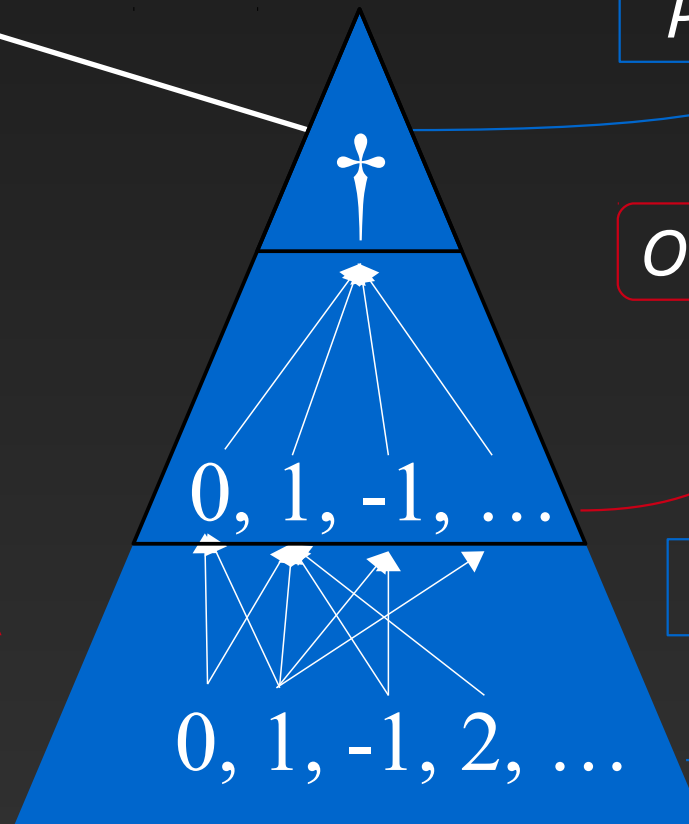
O answers



P answer

O questions

P answers



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

O question

P questions

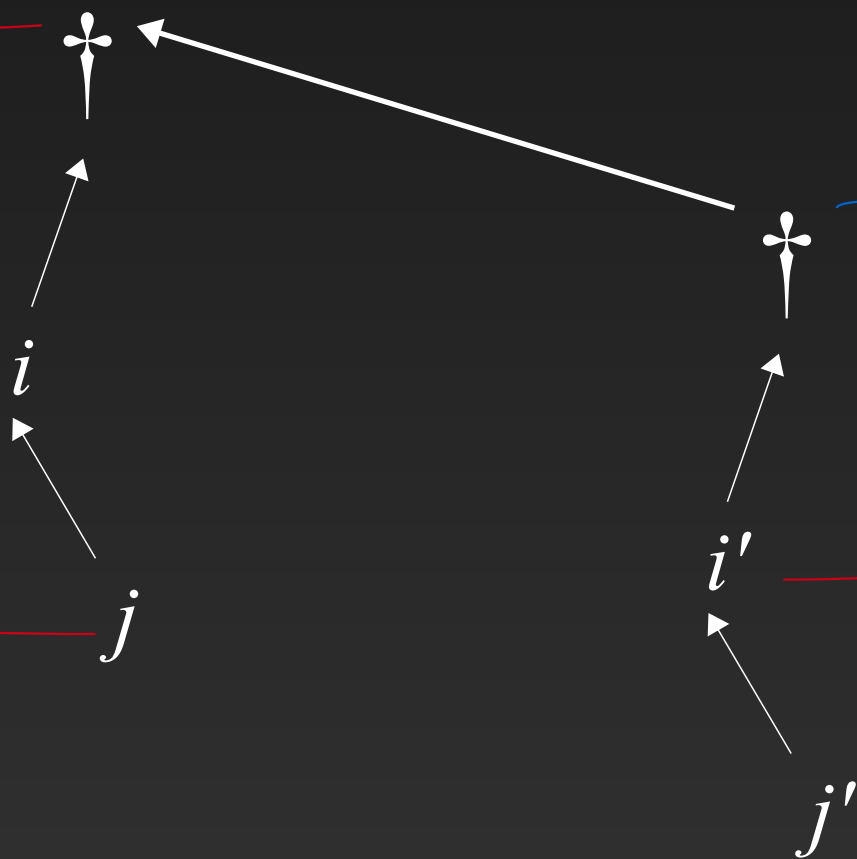
O answers

P answer

O questions

P answers

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



Example games

$f : \text{int} \rightarrow \text{int} \vdash \lambda x. f(x+37) : \text{int} \rightarrow \text{int}$

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

\dagger_f

O, Q

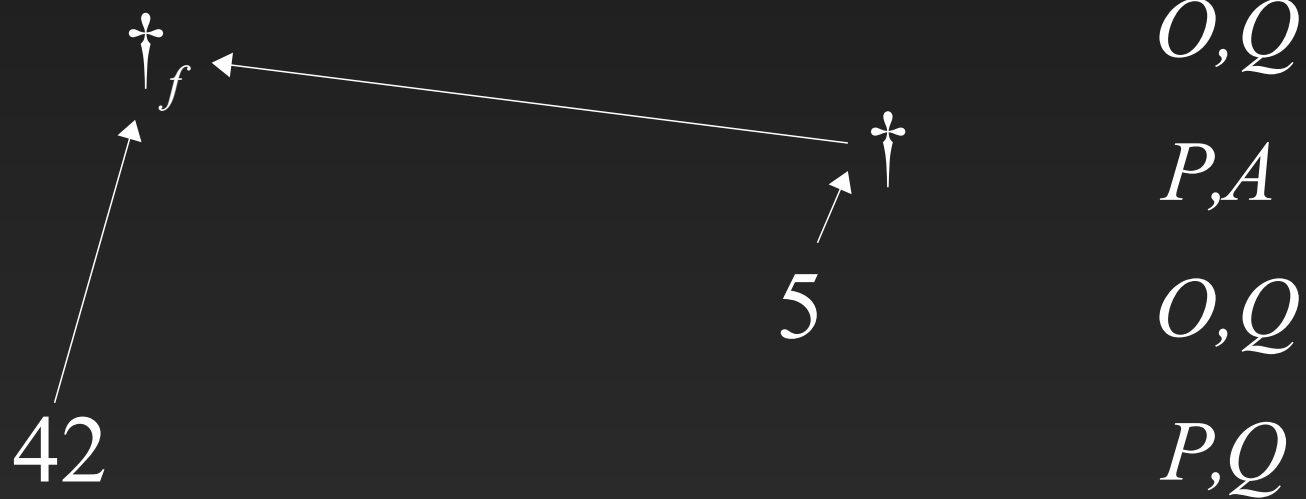
\dagger

P, A

Example games

$f : \text{int} \rightarrow \text{int} \vdash \lambda x. f(x+37) : \text{int} \rightarrow \text{int}$

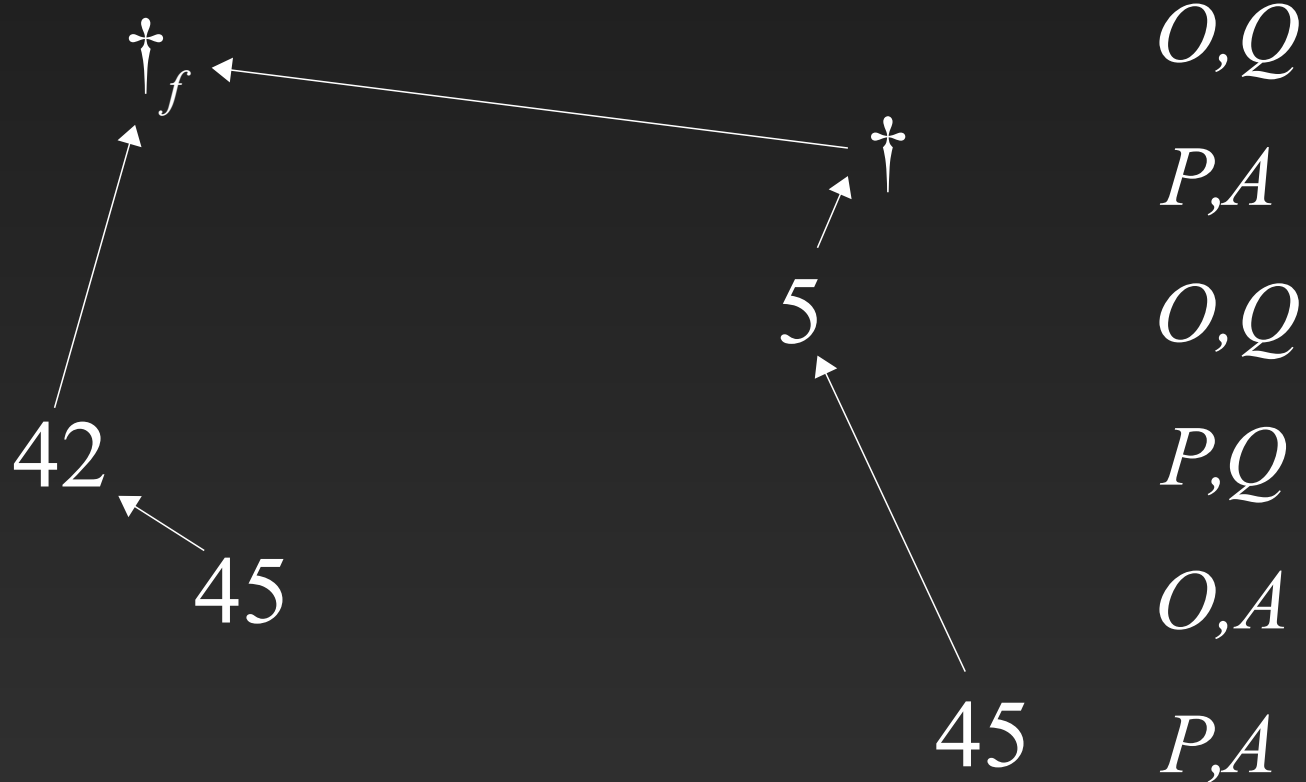
$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$



Example games

$f : \text{int} \rightarrow \text{int} \vdash \lambda x. f(x+37) : \text{int} \rightarrow \text{int}$

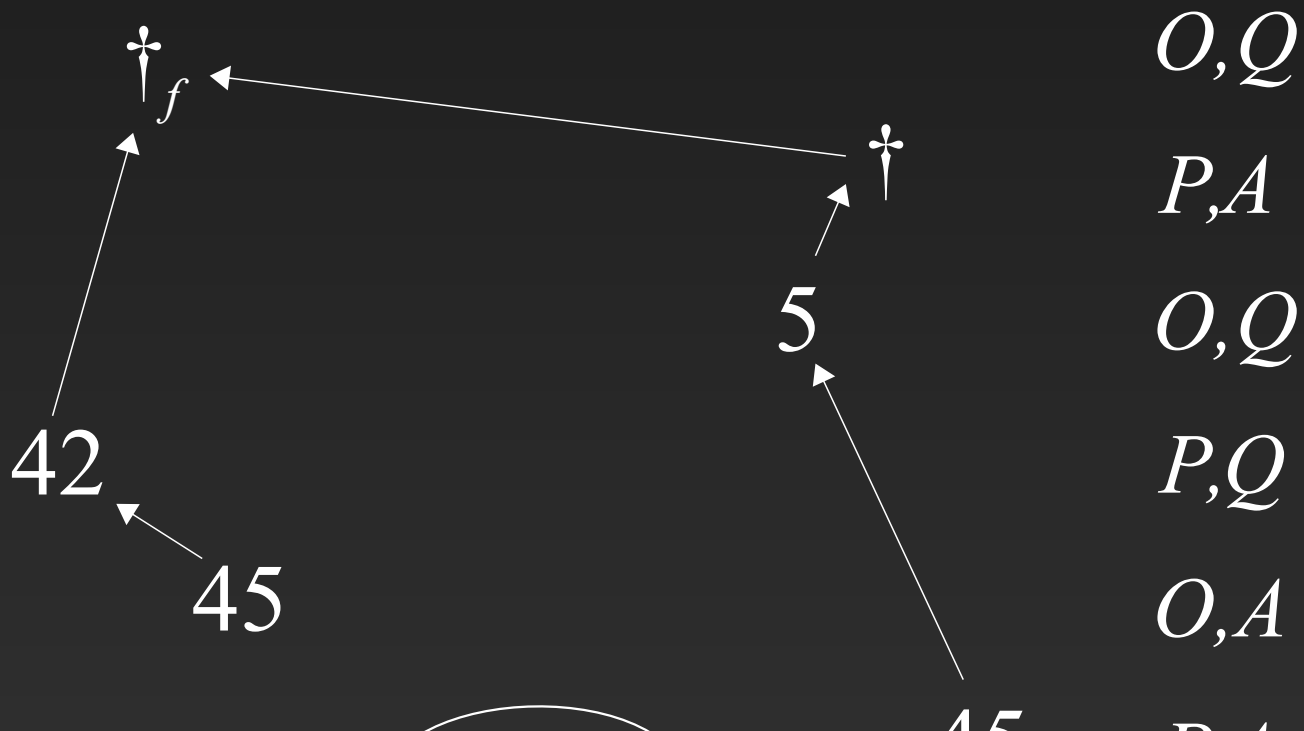
$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$



Example games

$f : \text{int} \rightarrow \text{int} \vdash \lambda x. f(x+37) : \text{int} \rightarrow \text{int}$

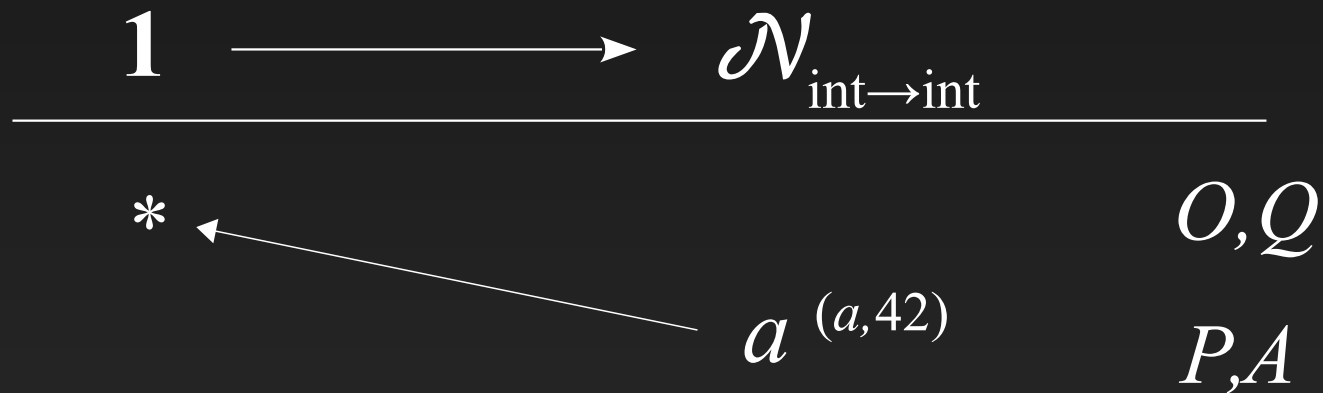
$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$



$$\llbracket \lambda x. f(x+37) \rrbracket = \{ \dagger_f \dagger i (i+37) j j \dots \}$$

Nominal games

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

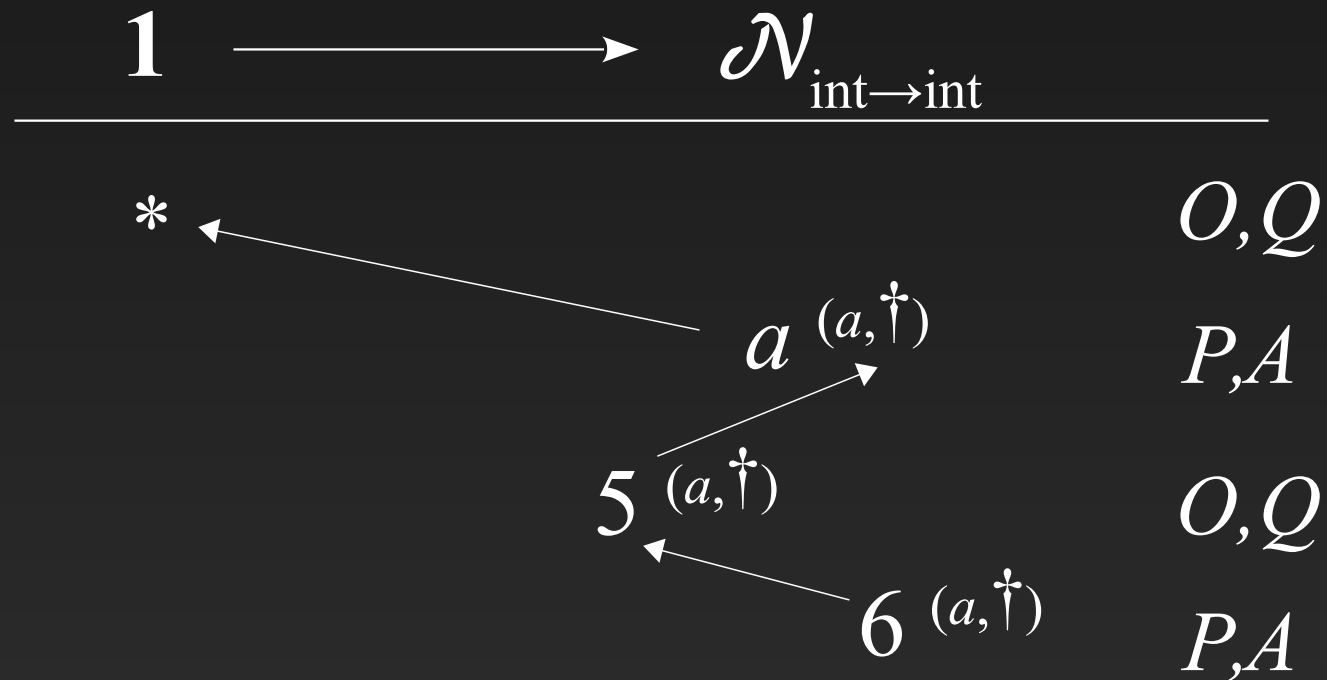


$$[[\text{ref } 42]] = \left\{ \begin{array}{c} \begin{array}{c} \curvearrowright \\ * \end{array} \\ a^{(a,42)} \\ \end{array} \right\}$$

$OQ \quad PA$

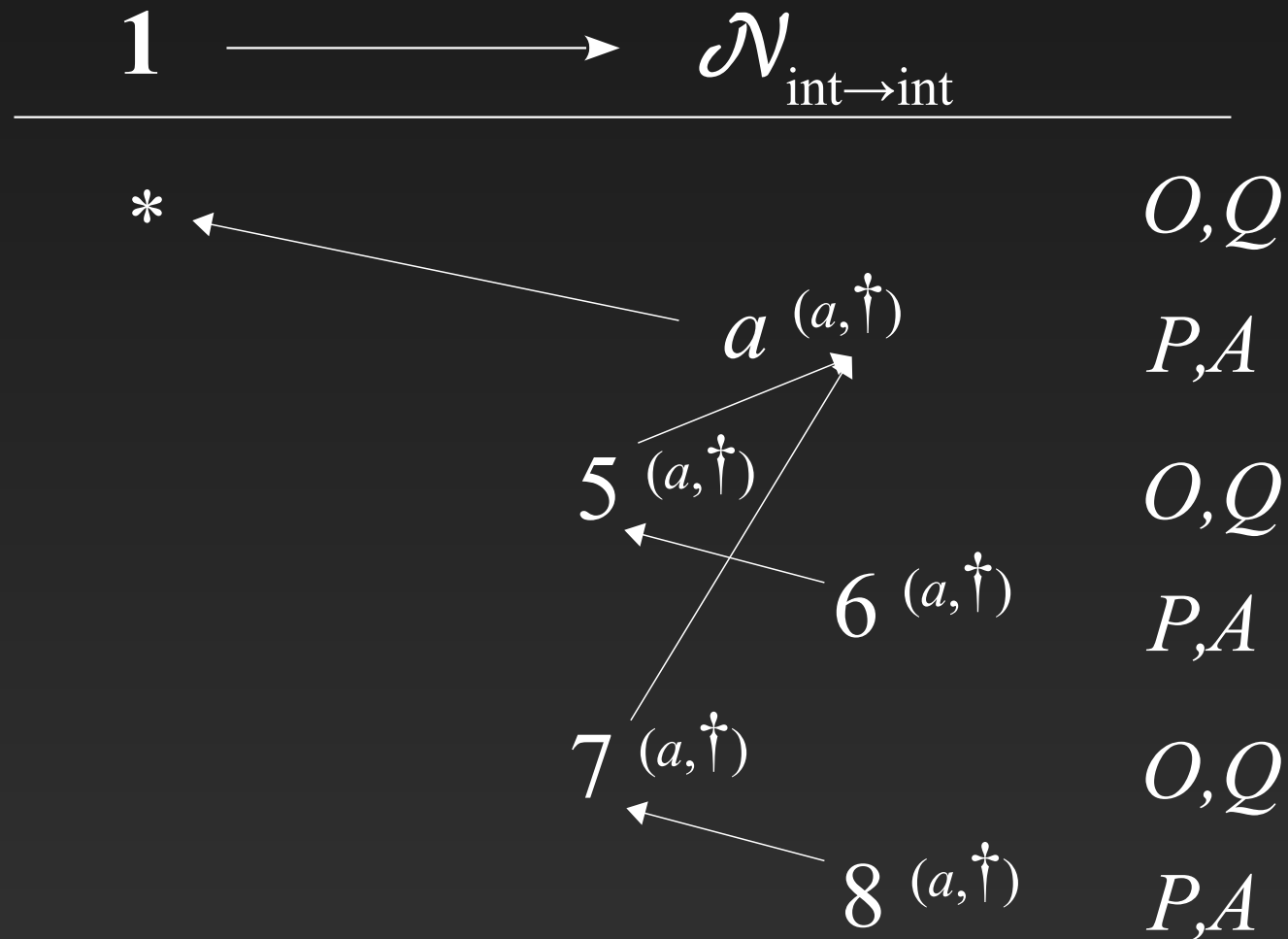
Nominal games

$\vdash \text{ref } (\lambda x^{\text{int}}.x+1) : \text{ref } (\text{int} \rightarrow \text{int})$



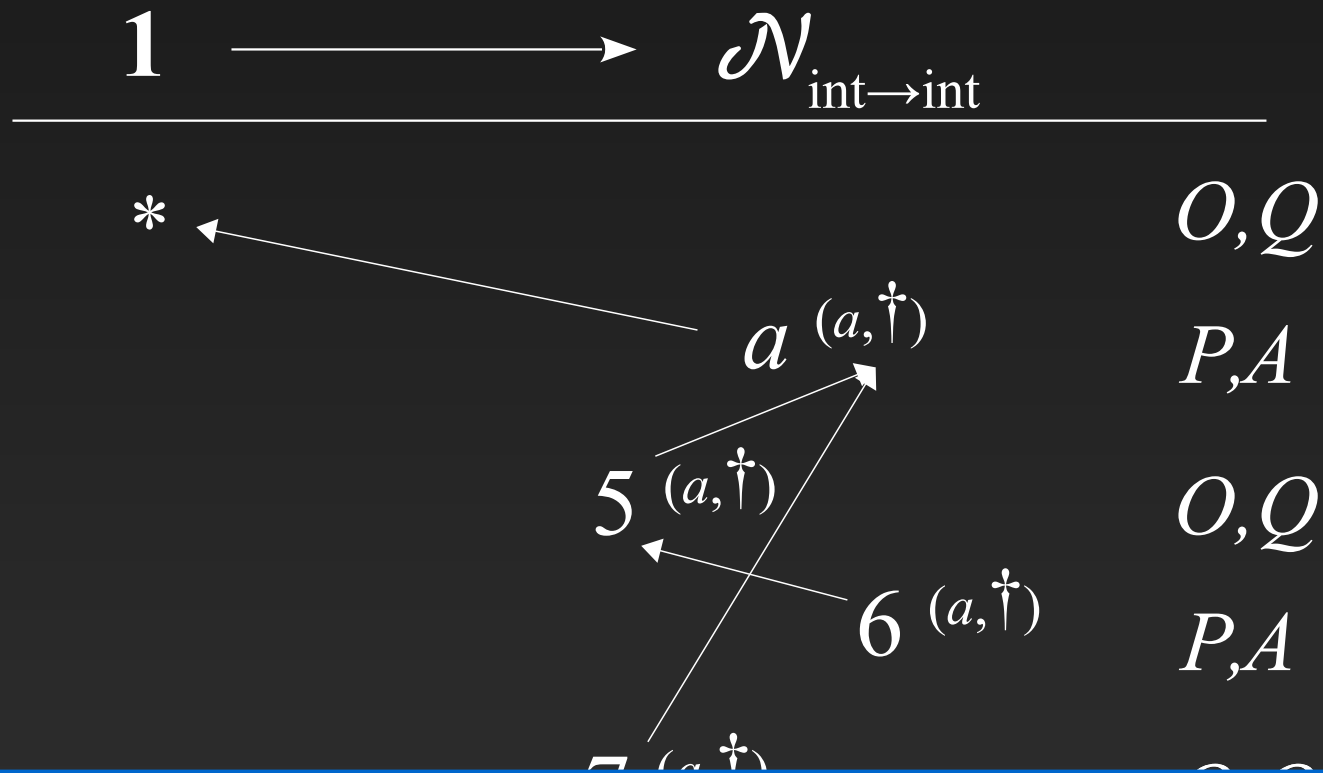
Nominal games

$\vdash \text{ref } (\lambda x^{\text{int}}.x+1) : \text{ref } (\text{int} \rightarrow \text{int})$



Nominal games

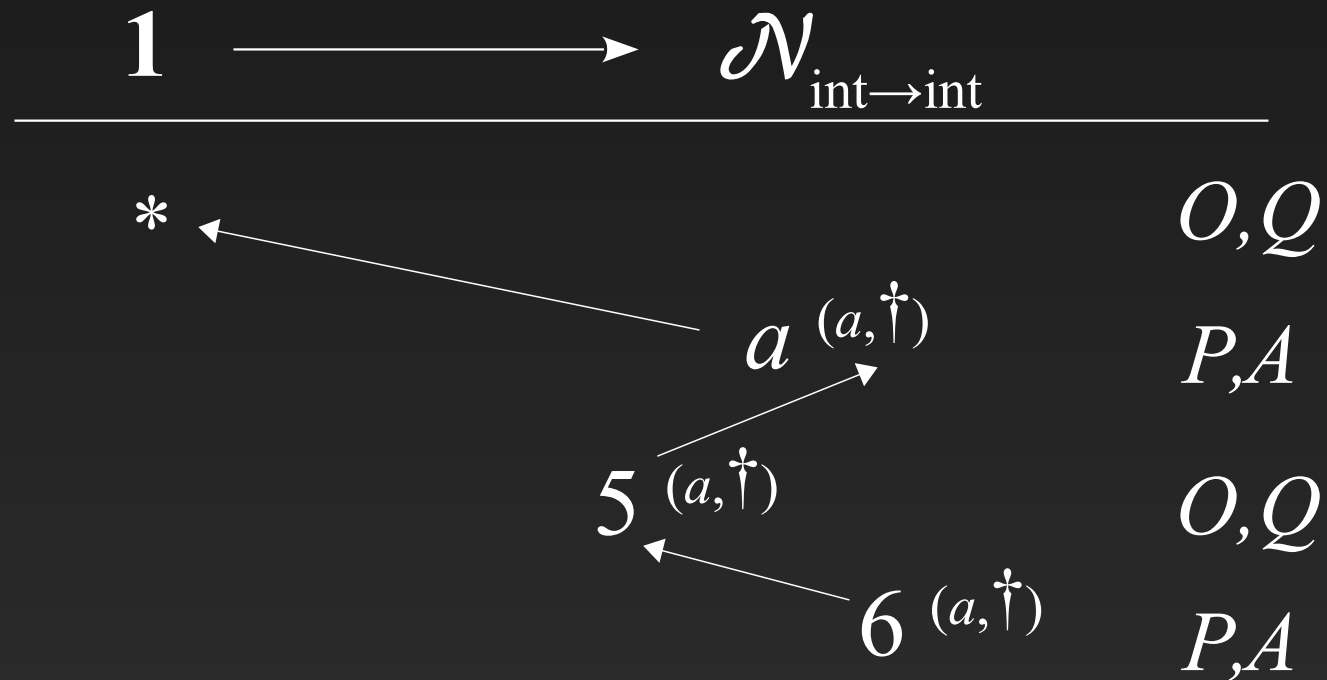
$\vdash \text{ref } (\lambda x^{\text{int}}.x+1) : \text{ref } (\text{int} \rightarrow \text{int})$



$$\llbracket \text{ref } \lambda x.x+1 \rrbracket = \left\{ \begin{array}{cccccc} * & a^{(a, \dagger)} & 5^{(a, \dagger)} & 6^{(a, \dagger)} & 7^{(a, \dagger)} & 8^{(a, \dagger)} & \dots \\ OQ & PA & OQ & PA & OQ & PA & \dots \end{array} \right\}$$

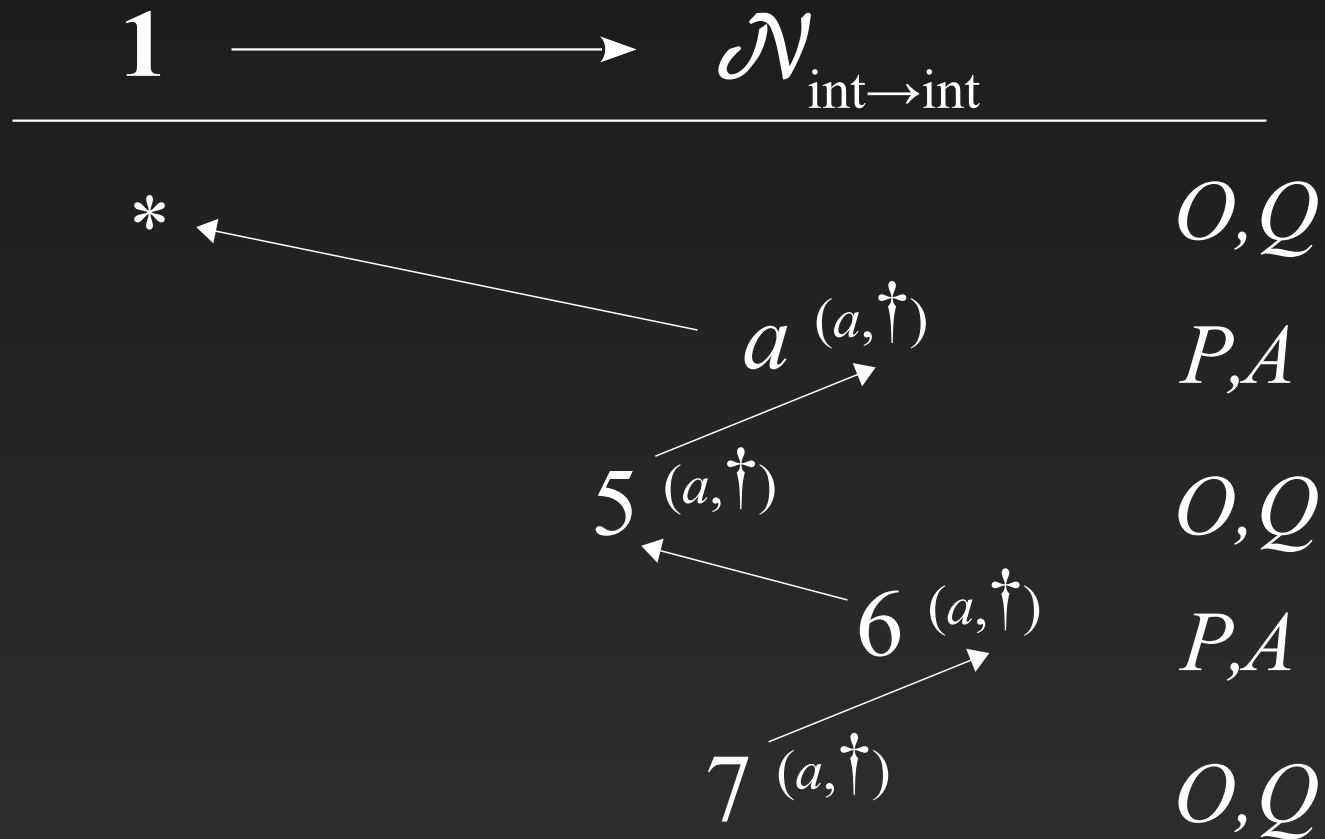
Nominal games

$\vdash \text{ref } (\lambda x^{\text{int}}.x+1) : \text{ref } (\text{int} \rightarrow \text{int})$



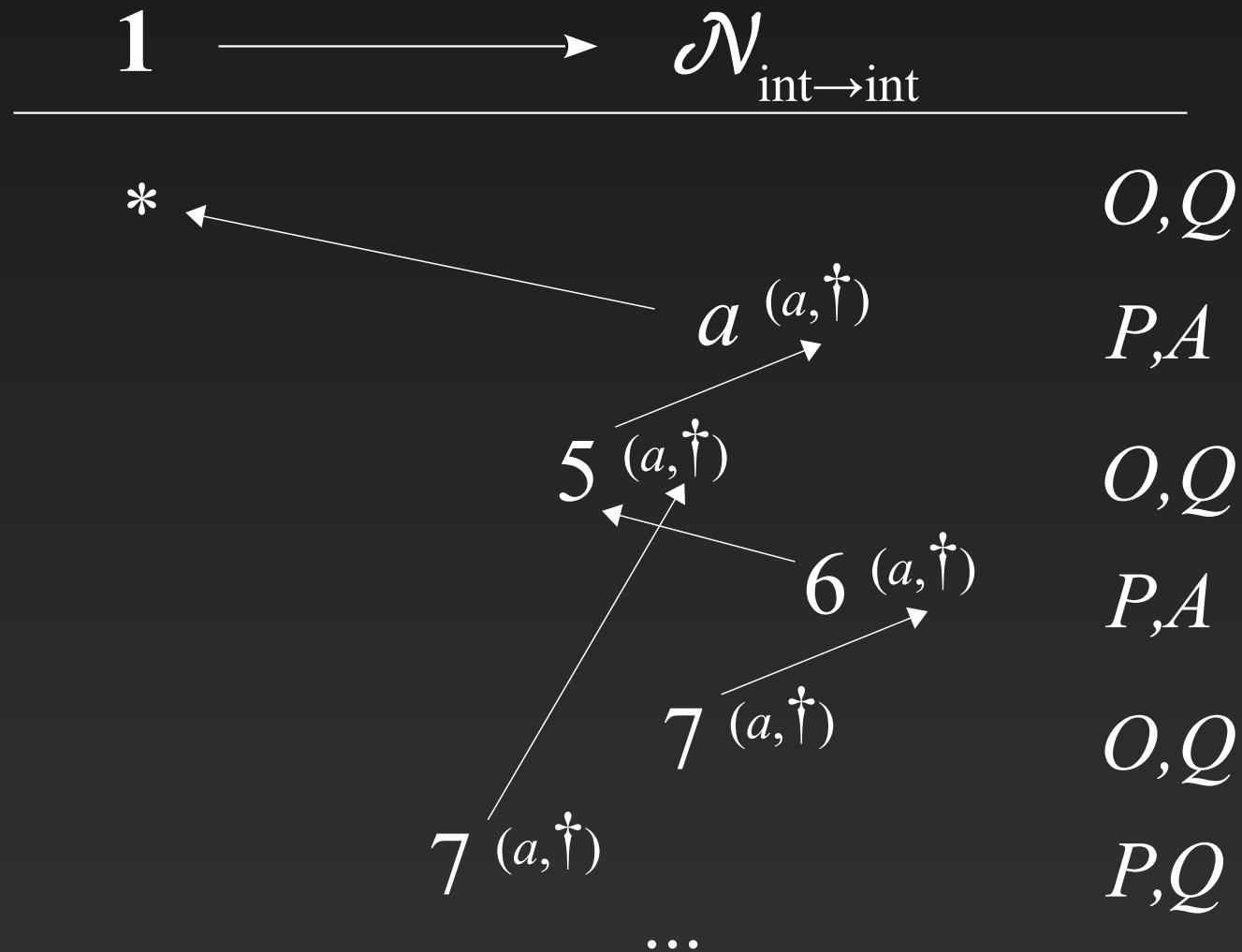
(there are more plays)

$\vdash \text{ref } (\lambda x^{\text{int}}.x+1) : \text{ref } (\text{int} \rightarrow \text{int})$



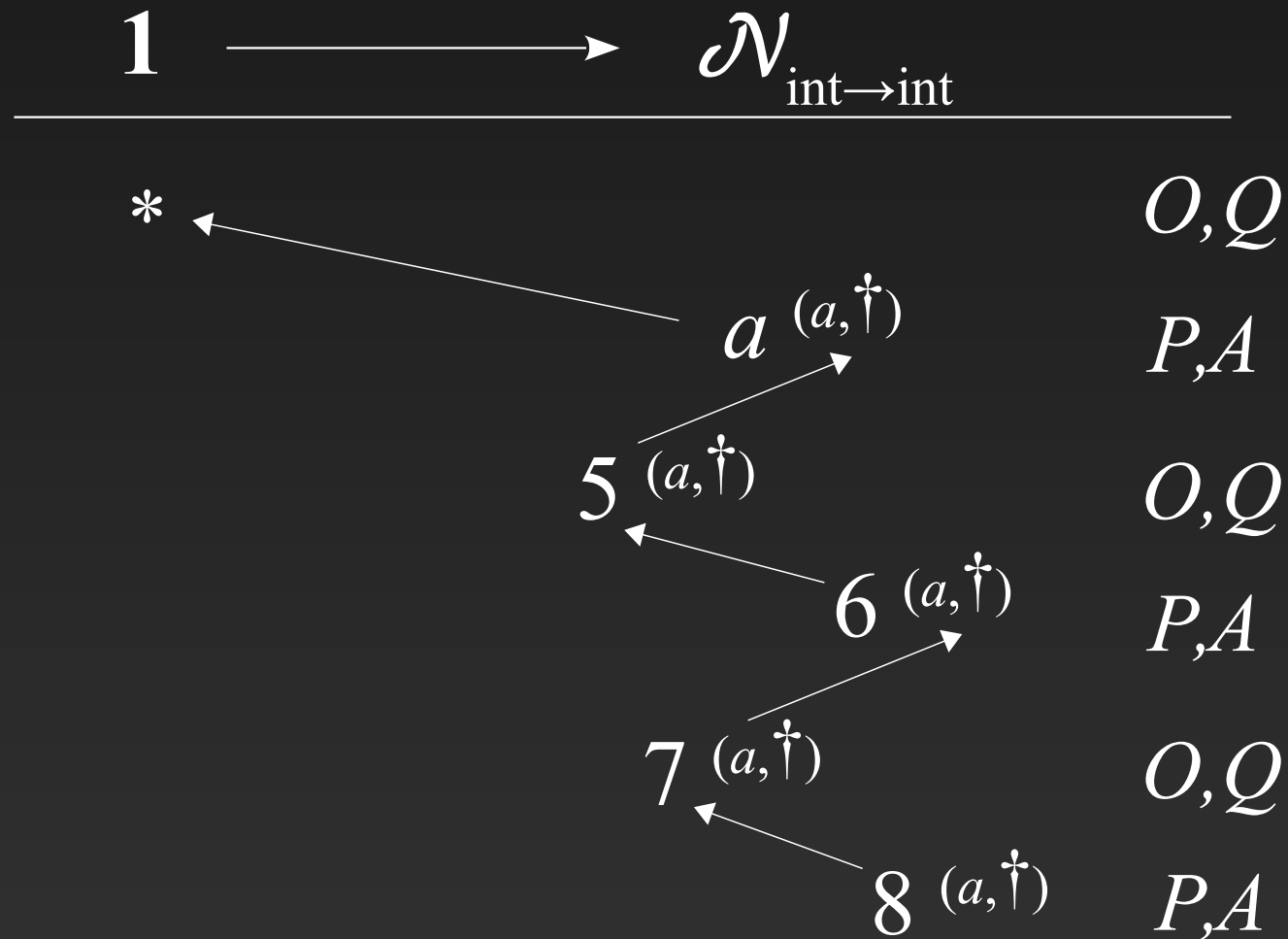
(there are more plays)

$\vdash \text{ref } (\lambda x^{\text{int}}.x+1) : \text{ref } (\text{int} \rightarrow \text{int})$



Quiz

$\vdash ??? : \text{ref}(\text{int} \rightarrow \text{int})$



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 $\llbracket \text{ref} \rrbracket$)
- prove correctness for each computational effect
- prove completeness via definability

Models for references

Games with moves that carry names and stores:

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

- pointers to moves & stores
- name privacy made explicit

Models for references

Games with moves that carry names and stores:

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

- 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

More effects: exceptions

$$\begin{array}{c}
 \frac{}{\Gamma \vdash () : \text{unit}} \quad \frac{}{\Gamma \vdash i : \text{int}} \quad \frac{}{\Gamma \vdash \text{if}_{\vartheta} : \text{int} \rightarrow \vartheta \rightarrow \vartheta \rightarrow \vartheta} \quad \frac{a \in \text{Loc}_{\vartheta}}{\Gamma \vdash a : \text{ref } \vartheta} \\
 \\
 \frac{}{\Gamma, x : \vartheta \vdash x : \vartheta} \quad \frac{\Gamma, x : \vartheta \vdash M : \vartheta'}{\Gamma \vdash \lambda x^{\vartheta}. M : \vartheta \rightarrow \vartheta'} \quad \frac{\Gamma \vdash M : \vartheta \rightarrow \vartheta' \quad \Gamma \vdash N : \vartheta}{\Gamma \vdash MN : \vartheta'} \\
 \\
 \frac{\Gamma \vdash M, N : \text{ref } \vartheta}{\Gamma \vdash M = N : \text{int}} \quad \boxed{\vartheta, \vartheta' ::= \text{unit} \mid \text{int} \mid \vartheta \rightarrow \vartheta' \mid \text{ref } \vartheta} \\
 \\
 \frac{\Gamma \vdash M : \vartheta}{\Gamma \vdash \text{ref } M : \text{ref } \vartheta} \quad \frac{\Gamma \vdash M : \text{ref } \vartheta}{\Gamma \vdash !M : \vartheta} \quad \frac{\Gamma \vdash M : \text{ref } \vartheta \quad \Gamma \vdash N : \vartheta}{\Gamma \vdash M := N : \text{unit}}
 \end{array}$$

More effects: exceptions

(exception names)

$$\begin{array}{c}
 \frac{}{\Gamma \vdash () : \text{unit}} \quad \frac{}{\Gamma \vdash i : \text{int}} \quad \frac{}{\Gamma \vdash \text{if}_{\vartheta} : \text{int} \rightarrow \vartheta \rightarrow \vartheta \rightarrow \vartheta} \quad \frac{e \in \text{Exn}}{\Gamma \vdash e : \text{exn}} \\
 \\
 \frac{}{\Gamma, x : \vartheta \vdash x : \vartheta} \quad \frac{\Gamma, x : \vartheta \vdash M : \vartheta'}{\Gamma \vdash \lambda x^{\vartheta}. M : \vartheta \rightarrow \vartheta'} \quad \frac{\Gamma \vdash M : \vartheta \rightarrow \vartheta' \quad \Gamma \vdash N : \vartheta}{\Gamma \vdash MN : \vartheta'}
 \end{array}$$

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

$$\vartheta, \vartheta' ::= \text{unit} \mid \text{int} \mid \vartheta \rightarrow \vartheta' \mid \text{exn}$$

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

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

Operational semantics excerpt

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

S : a set of exceptions

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

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

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

$$\text{exn}(), S \rightarrow e, S \uplus \{e\}$$

$$(\lambda x.M)(\text{raise } e), S \rightarrow \text{raise } e, S$$

$$v \text{ handle } x \Rightarrow M, S \rightarrow M[v/x], S$$

$$(\text{raise } e) \text{ handle } x \Rightarrow M, S \rightarrow M[e/x], S$$

Operational semantics excerpt

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

S : a set of exceptions

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

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

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

$$\text{exn}(), S \rightarrow e, S \uplus \{e\}$$

$$(\lambda x.M)(\text{raise } e), S \rightarrow \text{raise } e, S$$

$$v \text{ handle } x \Rightarrow M, S \rightarrow M[v/x], S$$

$$(\text{raise } e) \text{ handle } x \Rightarrow M, S \rightarrow M[e/x], S$$

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

Model for exceptions

Cater for **exceptional answer** moves: $e!$

- can answer any open question
- model otherwise unchanged



Model for exceptions

Cater for **exceptional answer** moves: $e!$

- can answer any open question
- model otherwise unchanged



Private exceptions:

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

→ restrict by **fresh-exception propagation**

Polymorphism

$$\begin{array}{c}
 \frac{}{\Gamma \vdash () : \text{unit}} \quad \frac{}{\Gamma \vdash i : \text{int}} \quad \frac{}{\Gamma \vdash \text{if}_{\vartheta} : \text{int} \rightarrow \vartheta \rightarrow \vartheta \rightarrow \vartheta} \quad \frac{e \in \text{Exn}}{\Gamma \vdash e : \text{exn}} \\
 \\
 \frac{}{\Gamma, x : \vartheta \vdash x : \vartheta} \quad \frac{\Gamma, x : \vartheta \vdash M : \vartheta'}{\Gamma \vdash \lambda x^{\vartheta}. M : \vartheta \rightarrow \vartheta'} \quad \frac{\Gamma \vdash M : \vartheta \rightarrow \vartheta' \quad \Gamma \vdash N : \vartheta}{\Gamma \vdash MN : \vartheta'}
 \end{array}$$

$\vartheta, \vartheta' ::= \text{unit} \mid \text{int} \mid \text{exn} \mid \vartheta \rightarrow \vartheta'$

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

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

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

Polymorphism

$$\frac{}{\Gamma \vdash () : \text{unit}} \quad \frac{}{\Gamma \vdash i : \text{int}} \quad \frac{}{\Gamma \vdash \text{if}_{\vartheta} : \text{int} \rightarrow \vartheta \rightarrow \vartheta \rightarrow \vartheta}$$
$$\frac{}{\Gamma, x : \vartheta \vdash x : \vartheta} \quad \frac{\Gamma, x : \vartheta \vdash M : \vartheta'}{\Gamma \vdash \lambda x^{\vartheta}. M : \vartheta \rightarrow \vartheta'} \quad \frac{\Gamma \vdash M : \vartheta \rightarrow \vartheta' \quad \Gamma \vdash N : \vartheta}{\Gamma \vdash MN : \vartheta'}$$

$$\vartheta, \vartheta' ::= \text{unit} \mid \text{int} \mid \alpha \mid \vartheta \rightarrow \vartheta' \mid \forall \alpha. \vartheta$$

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

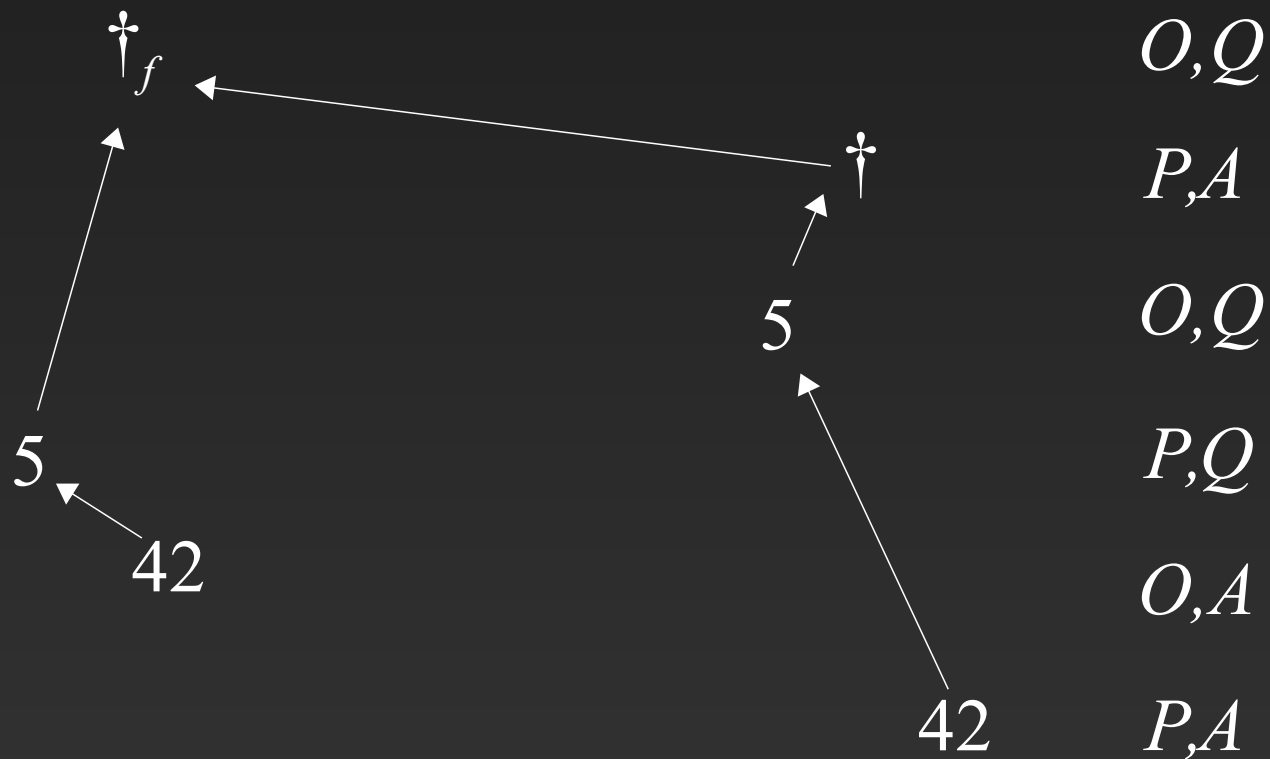
$$v ::= () \mid i \mid \text{if} \mid \Lambda \alpha. M \mid \lambda x. M$$

Model for polymorphism

Names for types and polymorphic values

$f : \text{int} \rightarrow \text{int} \vdash \lambda x. fx : \text{int} \rightarrow \text{int}$

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

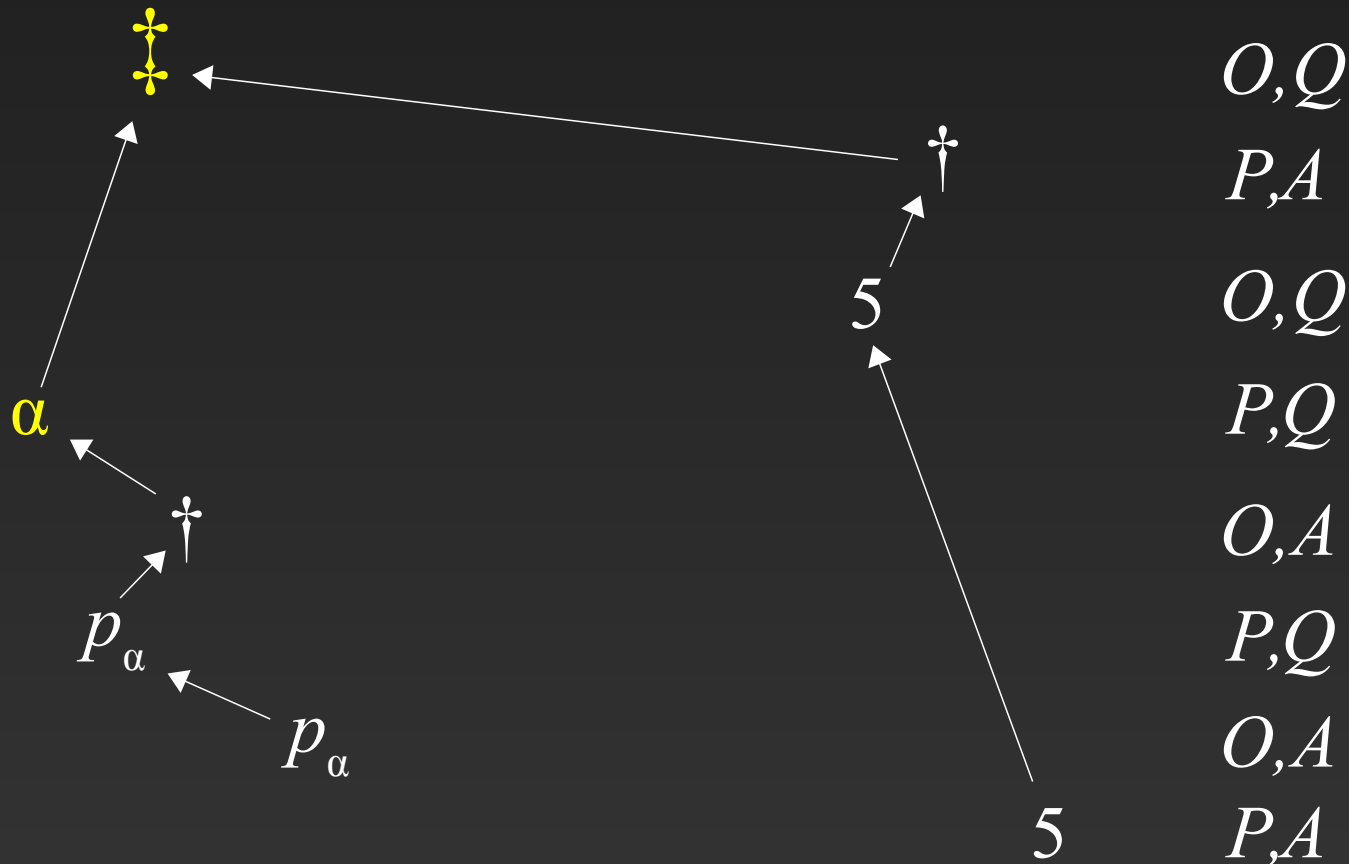


Model for polymorphism

Names for types and polymorphic values

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

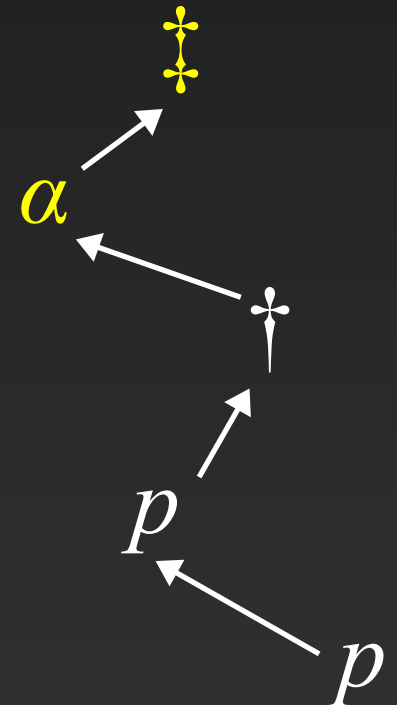
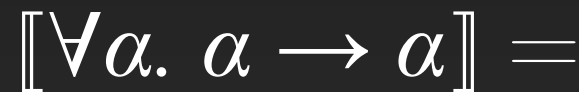
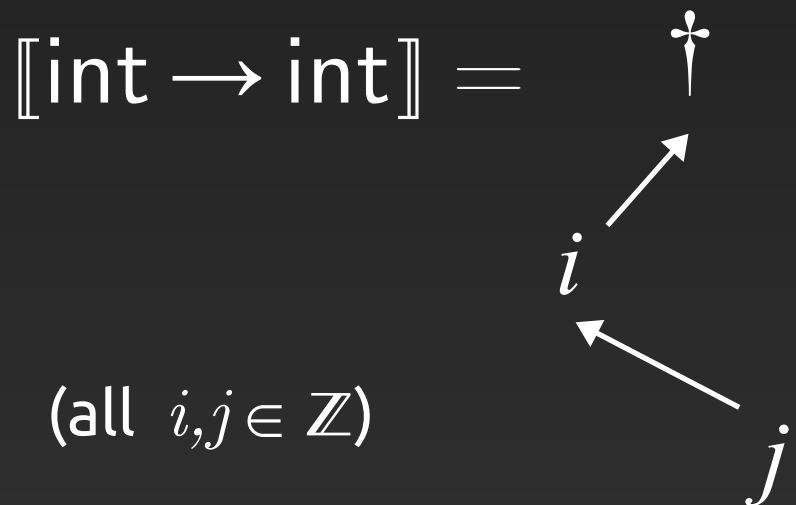
$\forall \alpha. \alpha \Rightarrow \alpha \quad \longrightarrow \quad \mathbb{Z} \Rightarrow \mathbb{Z}$



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*)



Names for:

- type variables

$$\alpha \in \mathcal{N}_{\text{TVar}}$$

- polymorphic values

$$p \in \mathcal{N}_{\alpha}$$

Games and traces

Games can be given a simpler, operational presentation:

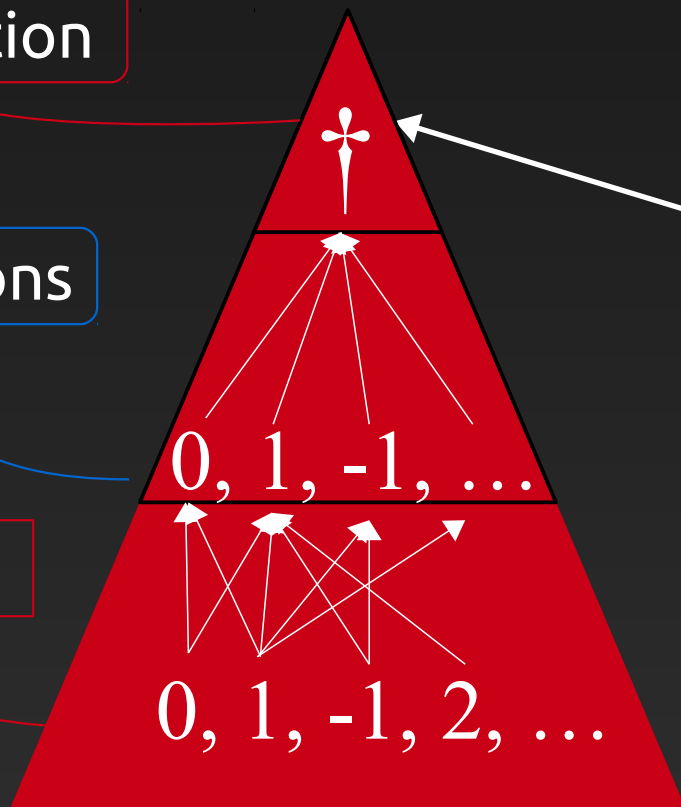
- pointer structure → named functions
- definition: denotational/compositional
→ operational/executable

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

O question

P questions

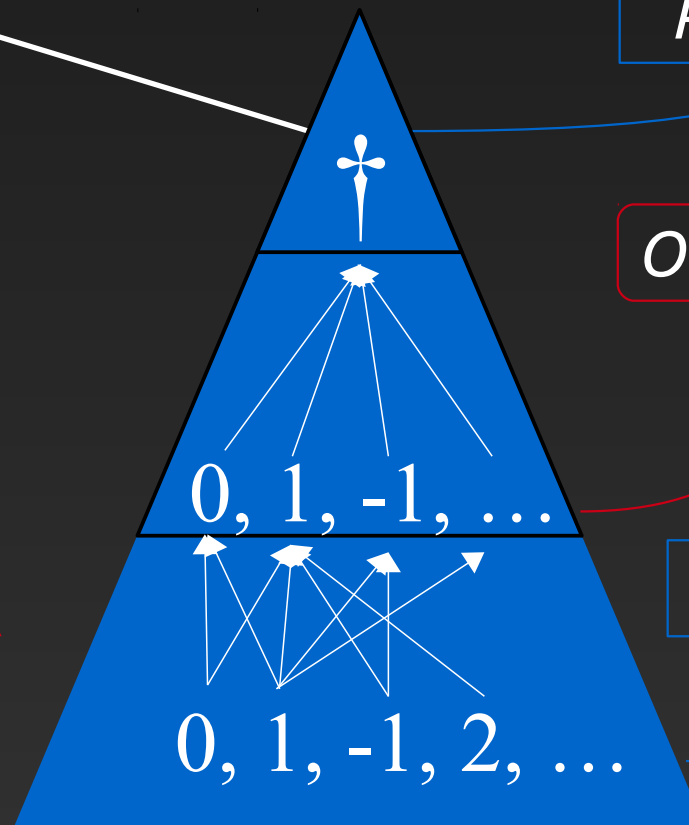
O answers



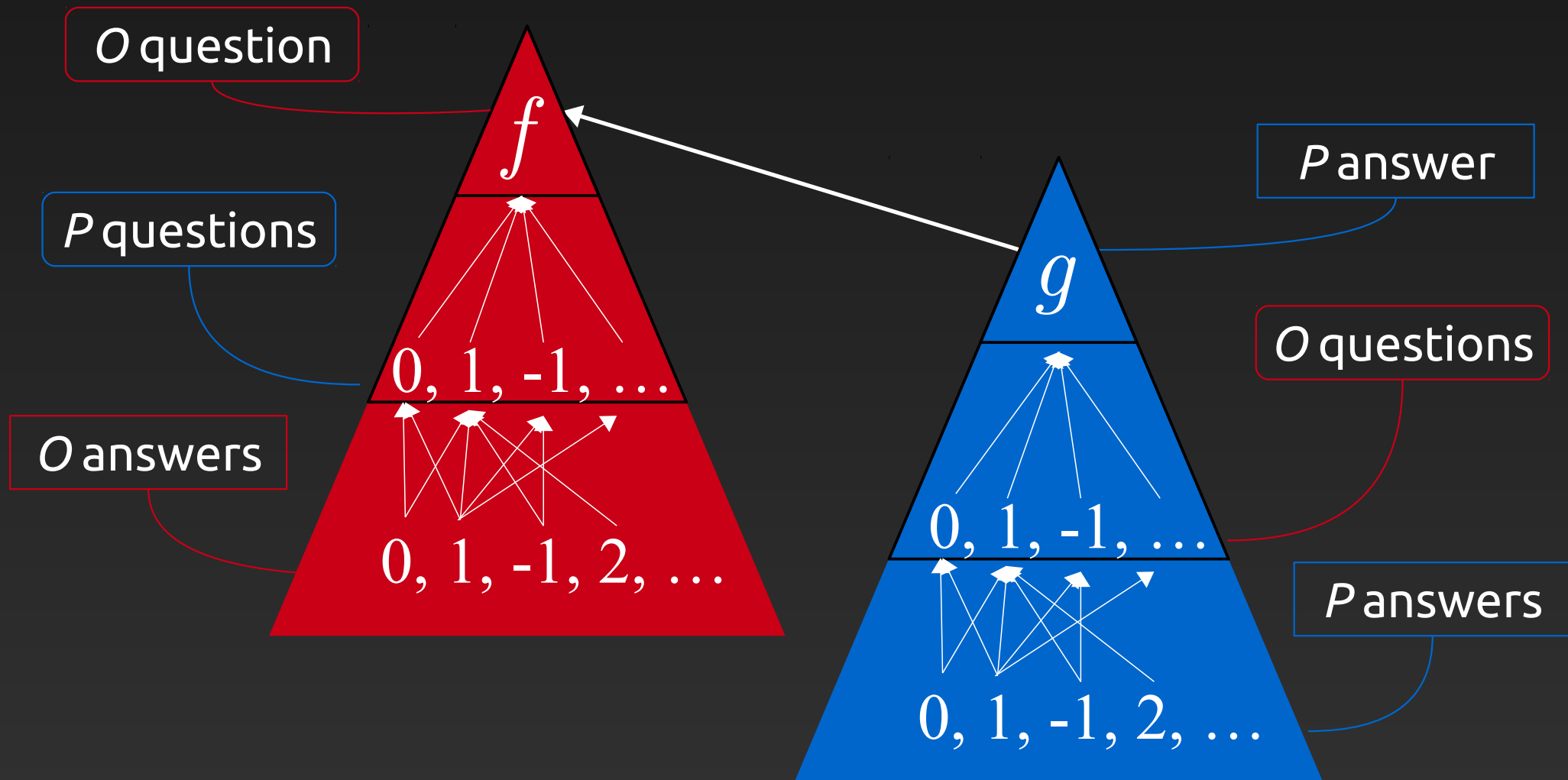
P answer

O questions

P answers

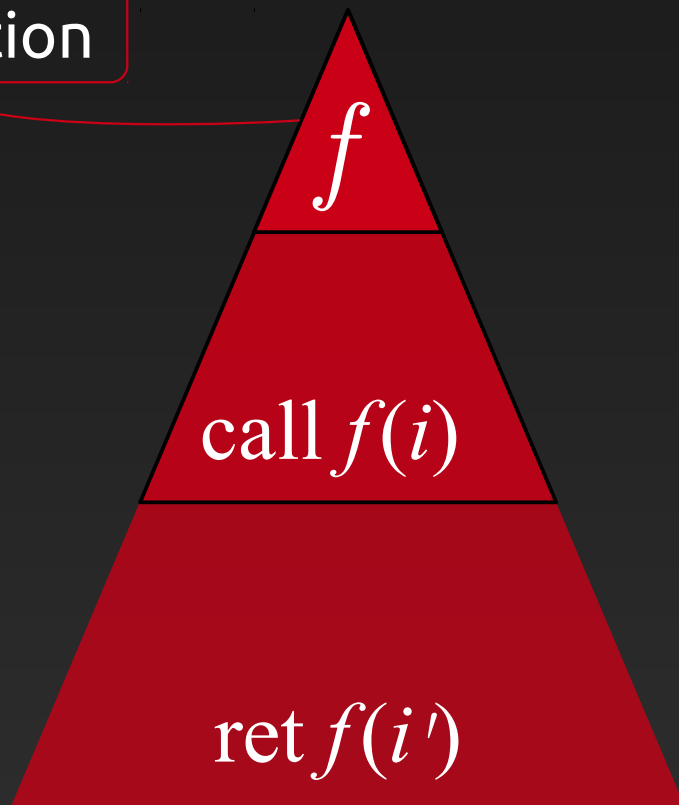


$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$

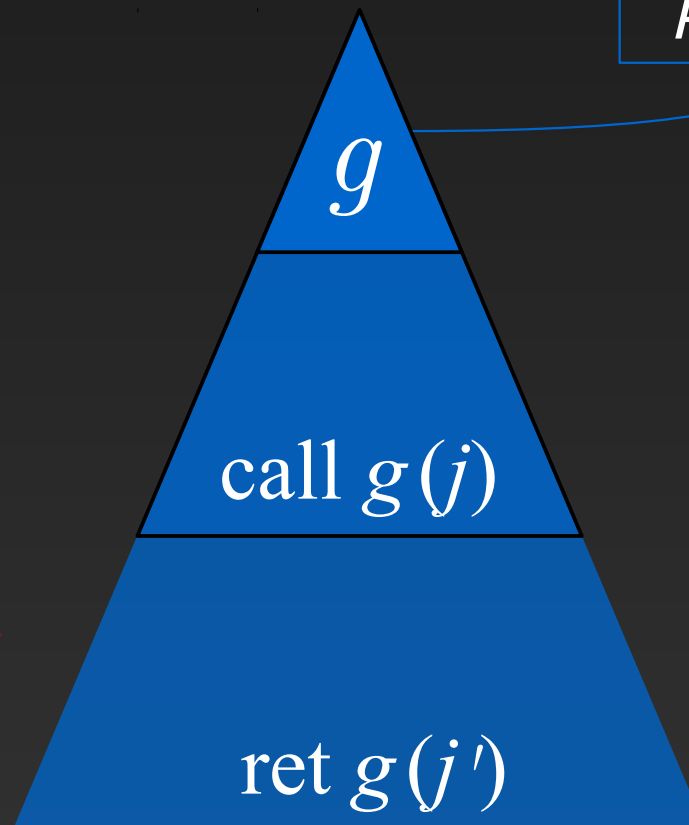


$$\mathbb{Z} \Rightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \Rightarrow \mathbb{Z}$$


O question



P answer



Example revisited

$$f : \text{int} \rightarrow \text{int} \vdash \lambda x. f(x+37) : \text{int} \rightarrow \text{int}$$
$$\llbracket \lambda x. f(x+37) \rrbracket = \{ \dagger_f \dagger_i \ (i+37) \ j \ j \ \dots \}$$

$$f \ g \ \text{call } g(i) \ \text{call } f(i+37) \ \text{ret } f(j) \ \text{ret } g(j) \ \dots$$

Operational games

Alan Jeffrey | Julian Rathke

Full Abstraction Factory

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*.
Copyright © 1994-2005 Alan Jeffrey and Julian Rathke. Edited 26 February 2007

[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*

Operational games

Defined operational, via an open transition system

- Configurations

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

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

- Transition rules

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

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!