Algorithmic Nominal Game Semantics

Nikos Tzevelekos

University of Oxford

Joint work with

Andrzej Murawski

University of Leicester

ESOP'11, Saarbrücken

What this talk is about

We present an automata-theoretic model of a finitary fragment of ML with integer references

The construction goes through game semantics and allows us to decide program equivalence

Ingredients

Reduced ML

Stark'95

Game Semantics

Murawski&Tz.'09

Fresh-Register Automata

Tz.'11

Ingredients

Reduced ML

Stark'95

Game Semantics

Murawski&Tz.'09

Fresh-Register Automata

Tz.'11

Algorithmic game semantics for Reduced ML

Reduced ML

θ ::= unit | int | intref | $\theta \to \theta$

Reduced ML

$$\theta \;::=\; \mathsf{unit} \mid \mathsf{int} \mid \mathsf{intref} \mid \theta \to \theta$$

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \qquad \frac{i \in \mathbb{Z}}{\Gamma \vdash i : \mathsf{int}} \qquad \frac{n \in \mathcal{N}}{\Gamma \vdash n : \mathsf{int\ ref}}$$

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

$$\frac{\Gamma \vdash M, N : \mathsf{int}}{\Gamma \vdash M \oplus N : \mathsf{int}} \qquad \frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash M_1, M_2 : \theta}{\Gamma \vdash \mathsf{if}\ M\ \mathsf{then}\ M_1\ \mathsf{else}\ M_0 : \theta} \qquad \frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash N : \mathsf{unit}}{\Gamma \vdash \mathsf{while}\ M\ \mathsf{do}\ N : \mathsf{unit}}$$

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

Reduced ML

names

$$\theta ::= \mathsf{unit} \mid \mathsf{int} \mid \mathsf{intref} \mid \theta \to \theta$$

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \qquad \frac{i \in \mathbb{Z}}{\Gamma \vdash i : \mathsf{int}} \qquad \boxed{\frac{n \in \mathcal{N}}{\Gamma \vdash n : \mathsf{int\ ref}}}$$

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

$$\frac{\Gamma \vdash M, N : \mathsf{int}}{\Gamma \vdash M \oplus N : \mathsf{int}} \qquad \frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash M_1, M_2 : \theta}{\Gamma \vdash \mathsf{if}\ M\ \mathsf{then}\ M_1\ \mathsf{else}\ M_0 : \theta} \qquad \frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash N : \mathsf{unit}}{\Gamma \vdash \mathsf{while}\ M\ \mathsf{do}\ N : \mathsf{unit}}$$

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

Reduced ML

names

$$\theta ::= \mathsf{unit} \mid \mathsf{int} \mid \mathsf{intref} \mid \theta \to \theta$$

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \qquad \frac{i \in \mathbb{Z}}{\Gamma \vdash i : \mathsf{int}} \qquad \frac{n \in \mathcal{N}}{\Gamma \vdash n : \mathsf{int\ ref}}$$

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

$$\frac{\Gamma \vdash M, N : \mathsf{int}}{\Gamma \vdash M \oplus N : \mathsf{int}} \qquad \frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash M_1, M_2 : \theta}{\Gamma \vdash \mathsf{if}\ M\ \mathsf{then}\ M_1\ \mathsf{else}\ M_0 : \theta} \qquad \frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash N : \mathsf{unit}}{\Gamma \vdash \mathsf{while}\ M\ \mathsf{do}\ N : \mathsf{unit}}$$

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

Reduced ML

$$s : \mathcal{N} \to_{\mathsf{fin}} \mathbb{Z}$$

Reduced ML

$$s : \mathcal{N} \to_{\mathsf{fin}} \mathbb{Z}$$

$$\frac{V \text{ is value}}{s, V \Downarrow s, V} \qquad \frac{M \Downarrow 0 \quad N_0 \Downarrow V}{\mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0 \Downarrow V} \qquad \frac{i \neq 0 \quad M \Downarrow i \quad N_1 \Downarrow V}{\mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0 \Downarrow V}$$

$$\frac{M_1 \Downarrow i_1 \quad M_2 \Downarrow i_2}{M_1 \oplus M_2 \Downarrow i_1 \oplus i_2} \qquad \frac{M \Downarrow \lambda x.M' \quad N \Downarrow V' \quad M'[V'/x] \Downarrow V}{M\,N \Downarrow V}$$

$$\frac{s, M \Downarrow s', i \quad n \notin \mathsf{dom}\, s'}{s, \mathsf{ref}\ M \Downarrow s' \uplus (n \mapsto i), n} \qquad \frac{s, M \Downarrow s', n \quad s'(n) = i}{s, !M \Downarrow s', i} \qquad \frac{s, M \Downarrow s', n \quad s', N \Downarrow s'', i}{s, M := N \Downarrow s''(n \mapsto i), ()}$$

Reduced ML

$$s : \mathcal{N} \to_{\mathsf{fin}} \mathbb{Z}$$

$$\frac{V \text{ is value}}{s, V \Downarrow s, V} \qquad \frac{M \Downarrow 0 \quad N_0 \Downarrow V}{\mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0 \Downarrow V} \qquad \frac{i \neq 0 \quad M \Downarrow i \quad N_1 \Downarrow V}{\mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0 \Downarrow V}$$

$$\frac{M_1 \Downarrow i_1 \quad M_2 \Downarrow i_2}{M_1 \oplus M_2 \Downarrow i_1 \oplus i_2} \qquad \frac{M \Downarrow \lambda x.M' \quad N \Downarrow V' \quad M'[V'/x] \Downarrow V}{M\,N \Downarrow V}$$

$$\frac{s, M \Downarrow s', i \quad n \notin \mathsf{dom}\, s'}{s, \mathsf{ref}\ M \Downarrow s' \uplus (n \mapsto i), n} \qquad \frac{s, M \Downarrow s', n \quad s'(n) = i}{s, !M \Downarrow s', i} \qquad \frac{s, M \Downarrow s', n \quad s', N \Downarrow s'', i}{s, M := N \Downarrow s''(n \mapsto i), ()}$$

Program equivalence

$\mathrm{M} \cong \mathrm{N}$ if, for all $C[\text{-}] : \texttt{unit}$, $C[\mathrm{M}]\Downarrow \iff C[\mathrm{N}]\Downarrow$

Program equivalence

$\mathrm{M} \cong \mathrm{N}$ if, for all $C[\text{-}] : \texttt{unit}$, $C[\mathrm{M}]\Downarrow \iff C[\mathrm{N}]\Downarrow$

$$x : \mathsf{int\ ref} \vdash x := 0; x := 1 \quad \cong \quad x := 1 : \mathsf{unit}$$

Program equivalence

$\mathrm{M} \cong \mathrm{N}$ if, for all $C[\text{-}] : \texttt{unit}$, $C[\mathrm{M}]\Downarrow \Leftrightarrow C[\mathrm{N}]\Downarrow$

$$x : \mathsf{int\ ref} \vdash x := 0; x := 1 \quad \cong \quad x := 1 : \mathsf{unit}$$

$$\mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\ \lambda y.\, (x == y) \quad \cong \quad \lambda y.\, 0 : \mathsf{int\ ref} \to \mathsf{int}$$

$$\mathsf{let}\, x = M\, \mathsf{in}\, N \ \equiv\ (\lambda x.N)M$$

Program equivalence

$\mathrm{M} \cong \mathrm{N}$ if, for all $C[\text{-}] : \texttt{unit}$, $C[\mathrm{M}]\Downarrow \iff C[\mathrm{N}]\Downarrow$

$x : \mathsf{int\ ref} \vdash x := 0; x := 1 \quad \cong \quad x := 1 : \mathsf{unit}$

$\mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\ \lambda y.\, (x == y) \quad \cong \quad \lambda y.\, 0 : \mathsf{int\ ref} \to \mathsf{int}$

$\lambda y.\, \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, x \quad \ncong \quad \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, (\lambda y.\, x) : \mathsf{unit} \to \mathsf{int\ ref}$

Program equivalence

$\mathrm{M} \cong \mathrm{N}$ if, for all $C[\text{-}] : \texttt{unit}$, $C[\mathrm{M}]\Downarrow \iff C[\mathrm{N}]\Downarrow$

$x : \mathsf{int\ ref} \vdash x := 0; x := 1 \quad \cong \quad x := 1 : \mathsf{unit}$

$\mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\ \lambda y.\, (x == y) \quad \cong \quad \lambda y.\, 0 : \mathsf{int\ ref} \to \mathsf{int}$

$\lambda y.\, \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, x \quad \ncong \quad \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, (\lambda y.\, x) : \mathsf{unit} \to \mathsf{int\ ref}$

$f : \mathsf{int\ ref} \to \mathsf{int} \ \vdash \ \lambda y.\, \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, f(x)$

$\cong \quad \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, \lambda y.\, x := 0; f(x) : \mathsf{unit} \to \mathsf{int}$

Program equivalence

$\mathrm{M} \cong \mathrm{N}$ if, for all $C[\text{-}] : \texttt{unit}$, $C[\mathrm{M}]\Downarrow \iff C[\mathrm{N}]\Downarrow$

$x : \mathsf{int\ ref} \vdash x := 0; x := 1 \quad \cong \quad x := 1 : \mathsf{unit}$

$\mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\ \lambda y.\, (x == y) \quad \cong \quad \lambda y.\, 0 : \mathsf{int\ ref} \to \mathsf{int}$

$\lambda y.\, \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, x \quad \ncong \quad \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, (\lambda y.\, x) : \mathsf{unit} \to \mathsf{int\ ref}$

$f : \mathsf{int\ ref} \to \mathsf{int} \ \vdash \ \lambda y.\, \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, f(x)$

$\cong \quad \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, \lambda y.\, x := 0; f(x) : \mathsf{unit} \to \mathsf{int}$

$f : \mathsf{int\ ref} \to \mathsf{unit} \ \vdash \ \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, \mathsf{let}\, y = \mathsf{ref}\ (0)\, \mathsf{in}\, f(x); (y := !x); y$

$\cong \quad \mathsf{let}\, x = \mathsf{ref}\, 0\, \mathsf{in}\, f(x); x : \mathsf{int\ ref}$

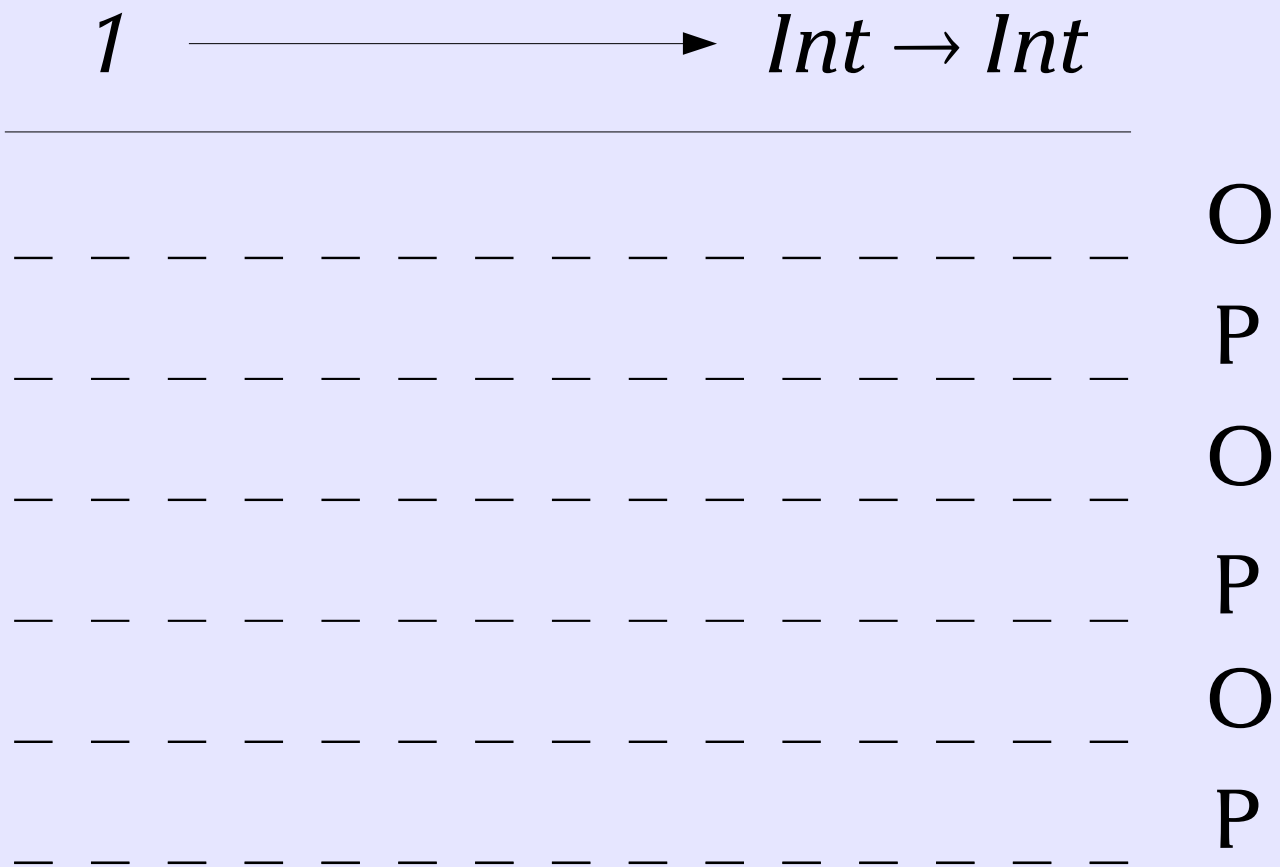
Game Semantics

Game Semantics

- Computation is modelled as a 2-player game between:
  - *Opponent* (the environment)
  - *Proponent* (the program)

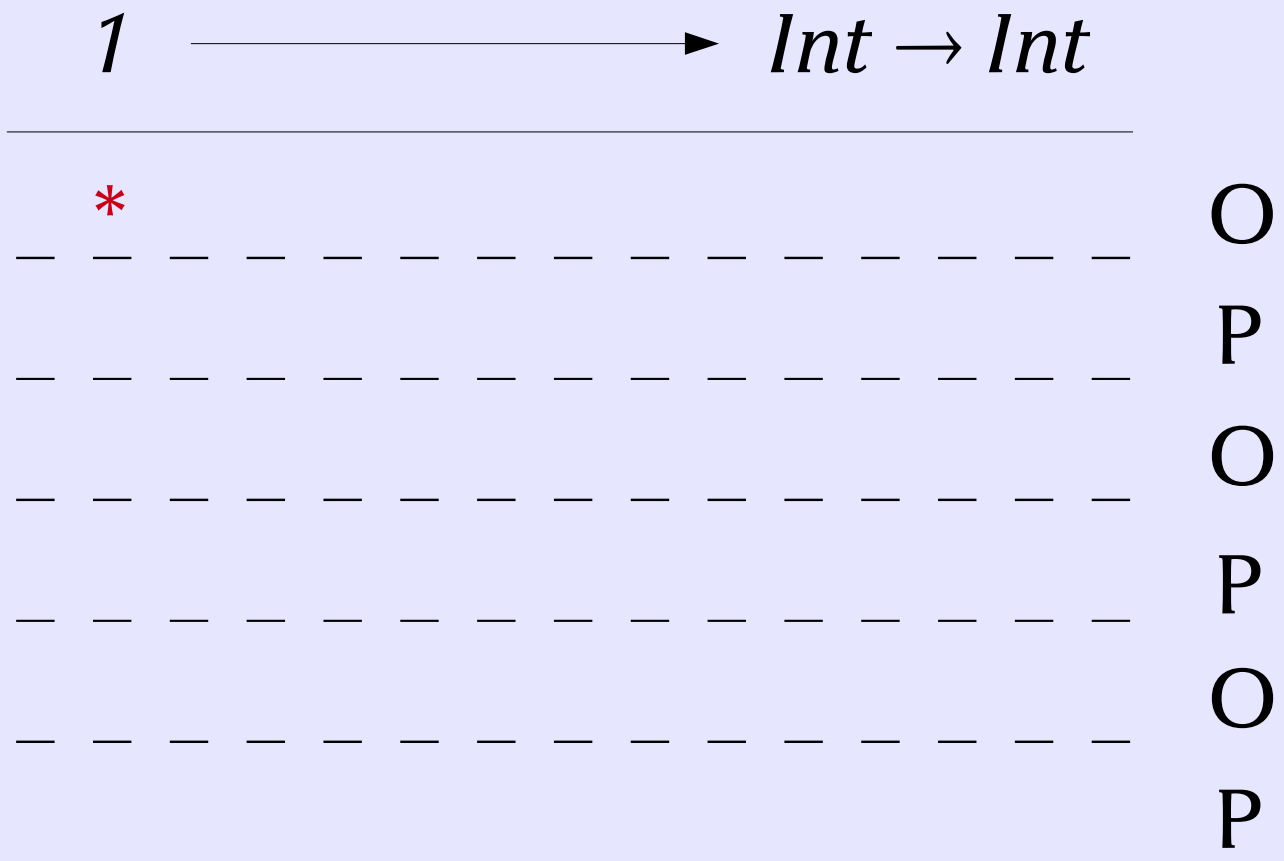
Examples

$\vdash$ `λx.x+1 : int → int`



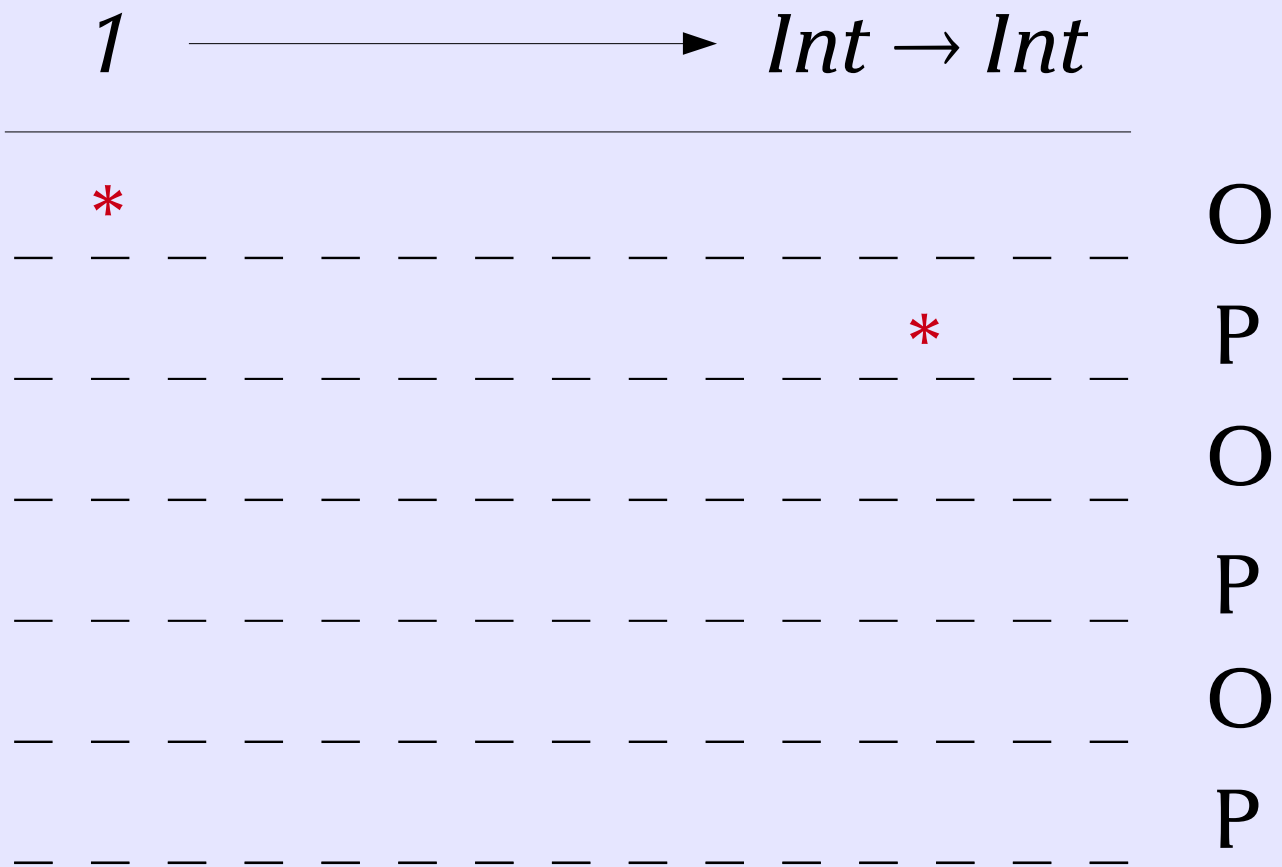
Examples

$\vdash$ `λx.x+1 : int → int`



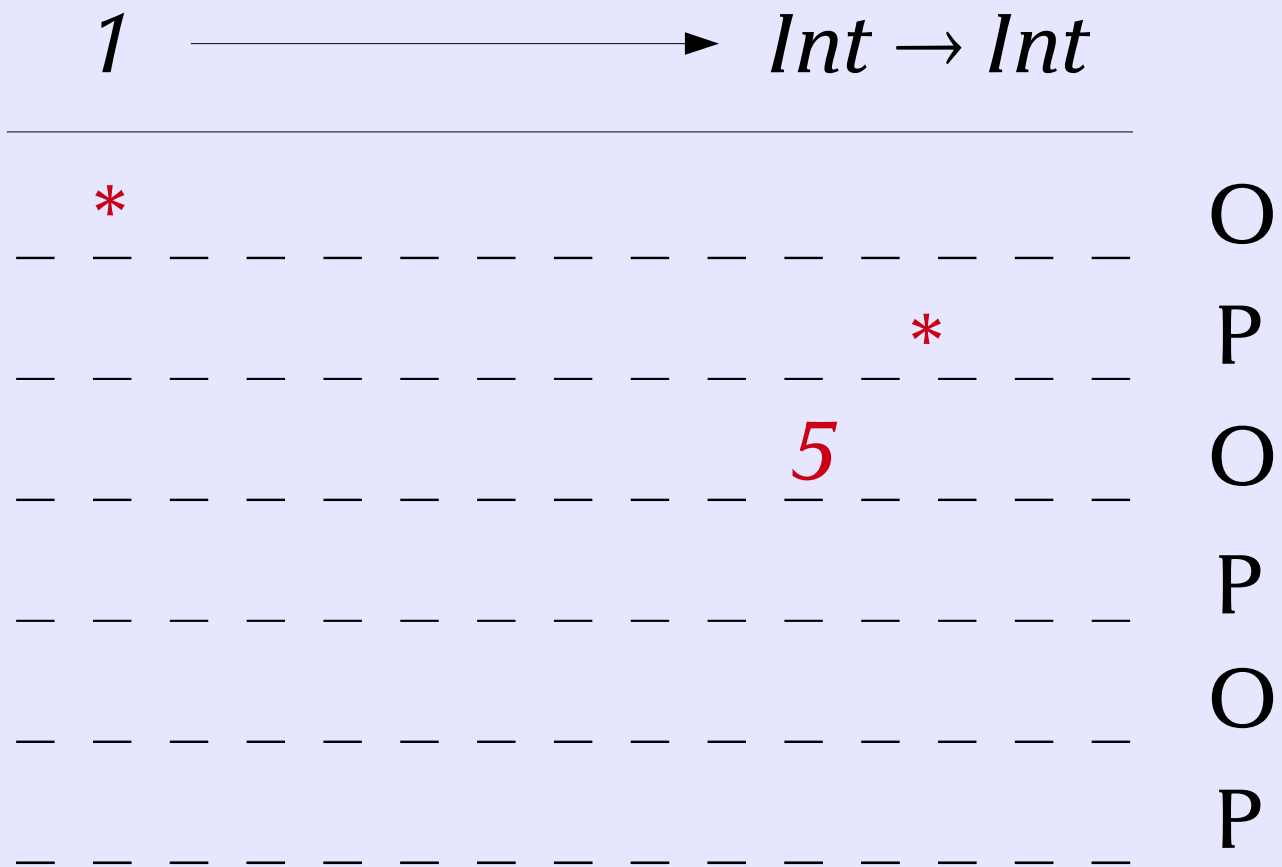
Examples

$\vdash$ `λx.x+1 : int → int`



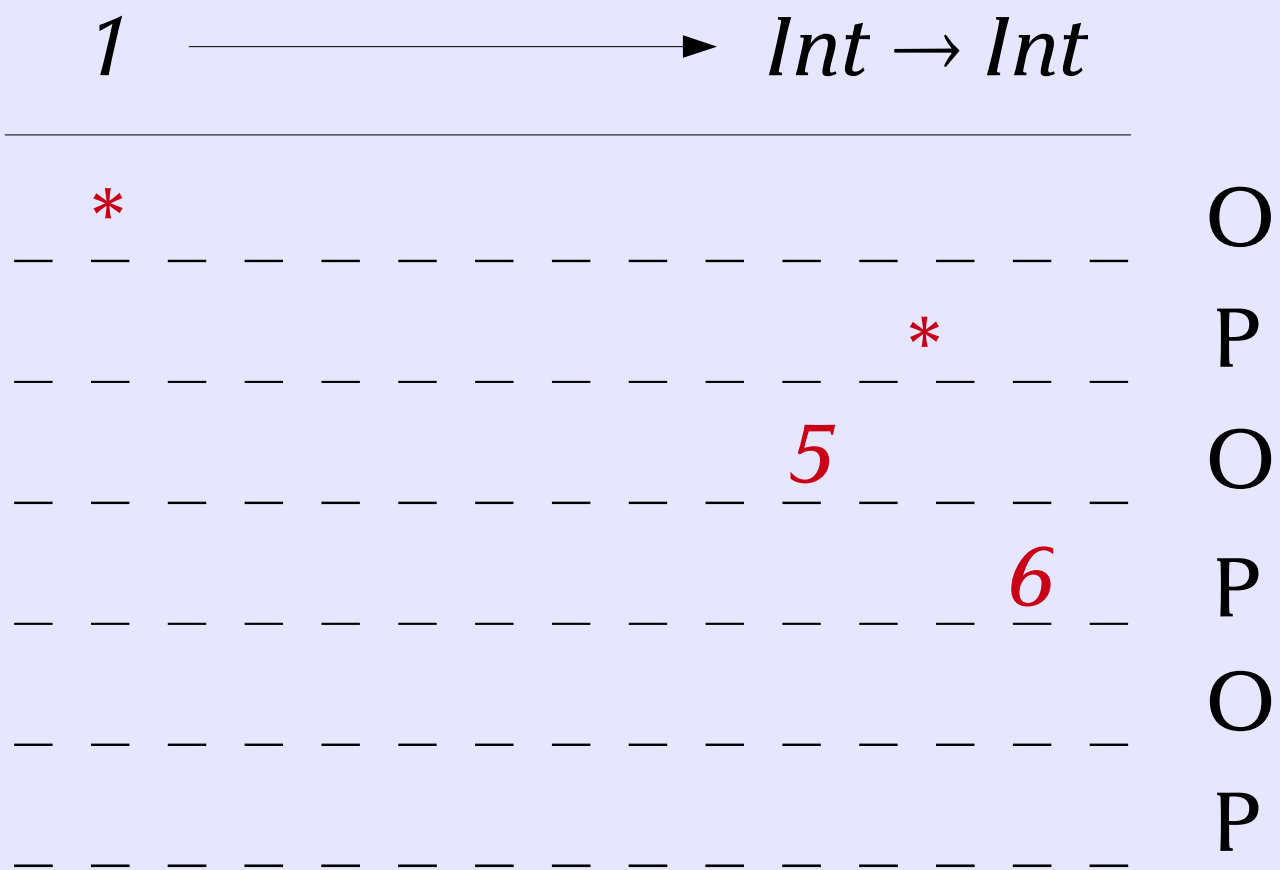
Examples

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



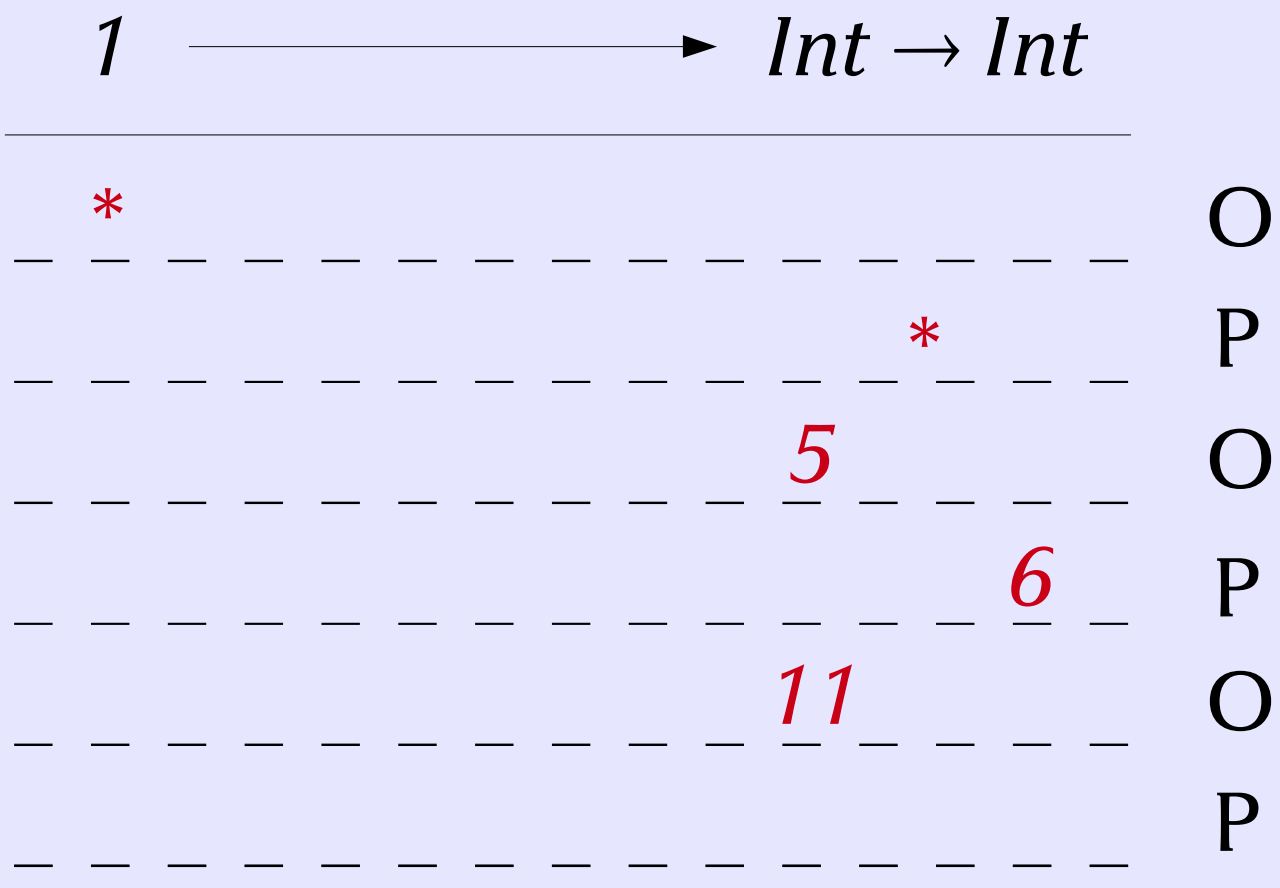
Examples

$\vdash$ `λx.x+1 : int → int`



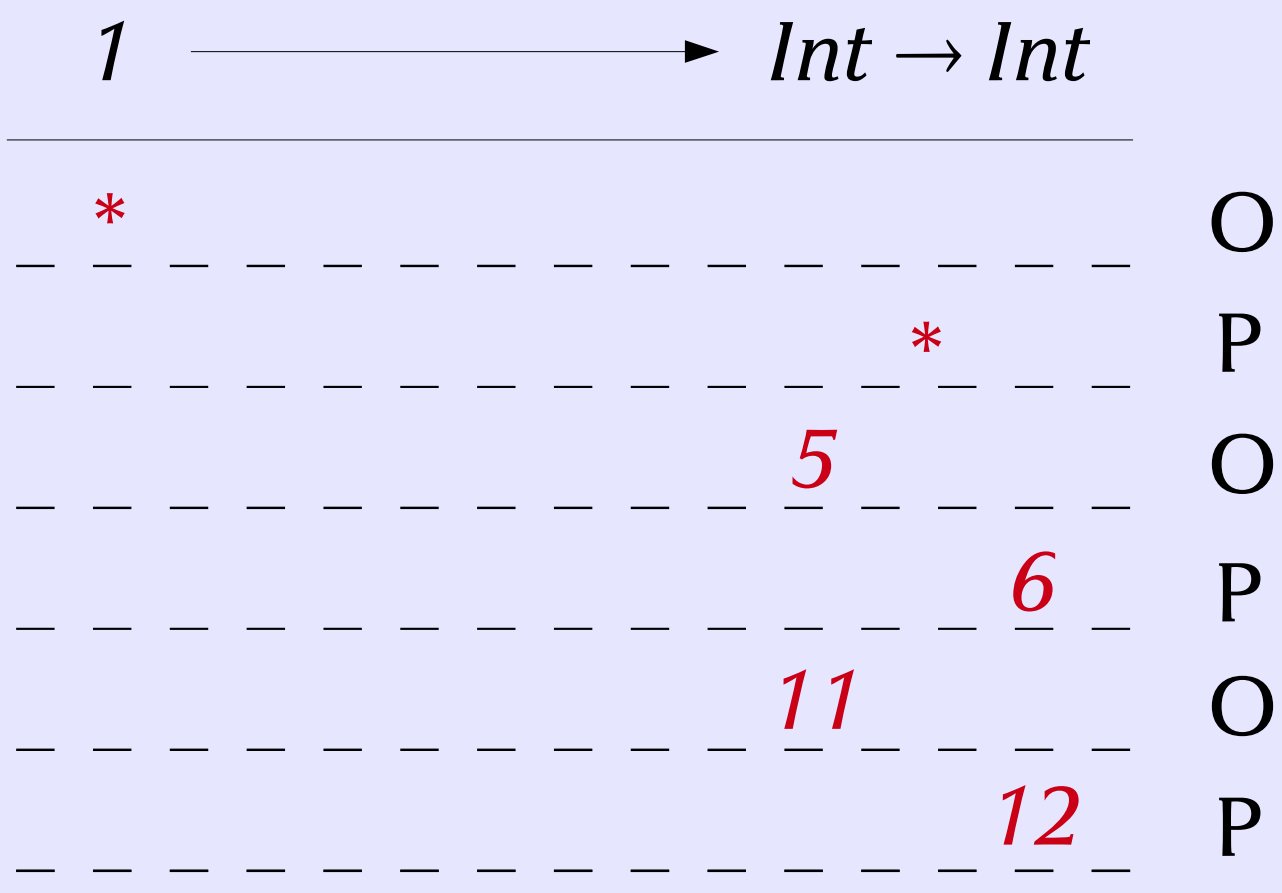
Examples

$\vdash$ `λx.x+1 : int → int`



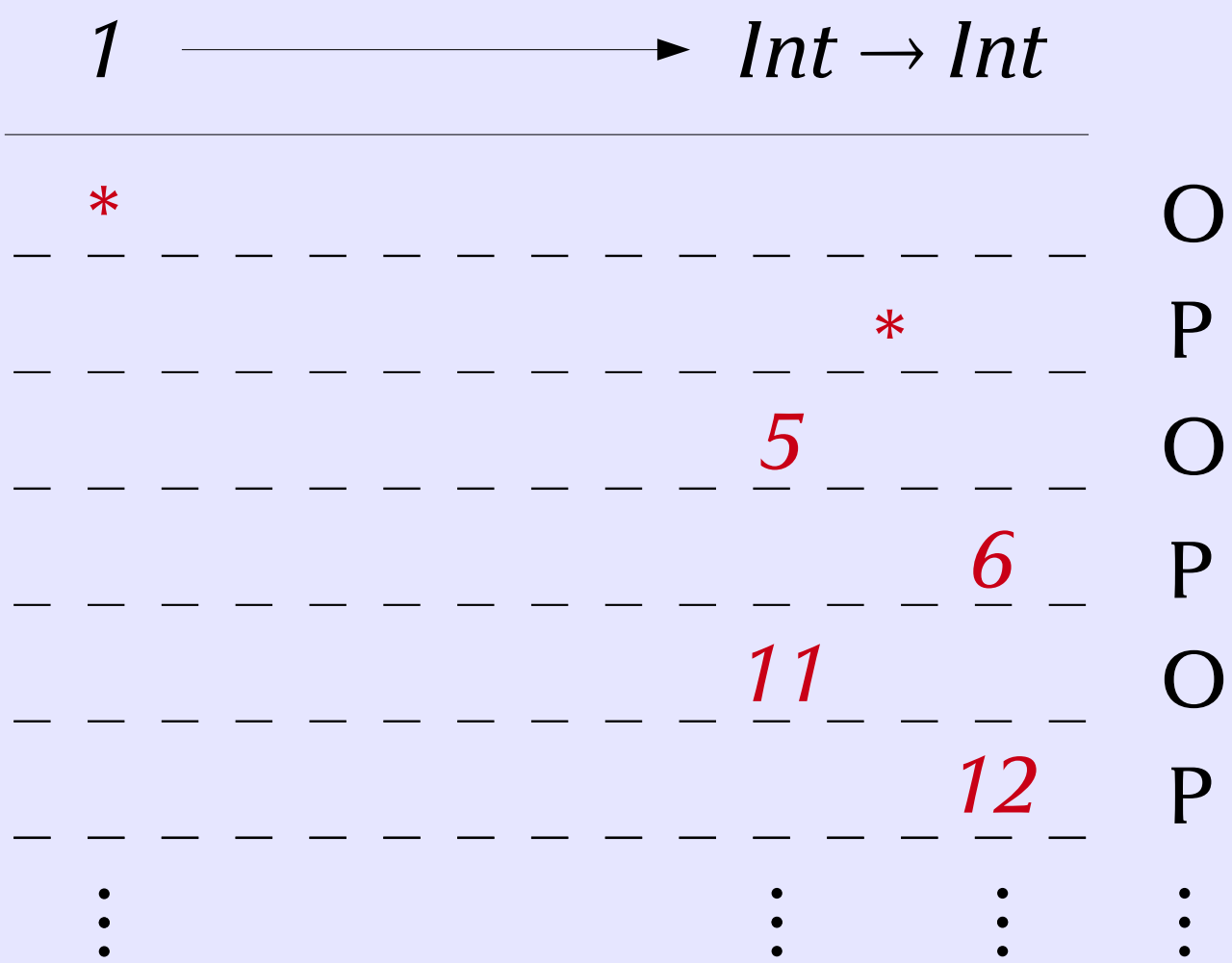
Examples

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



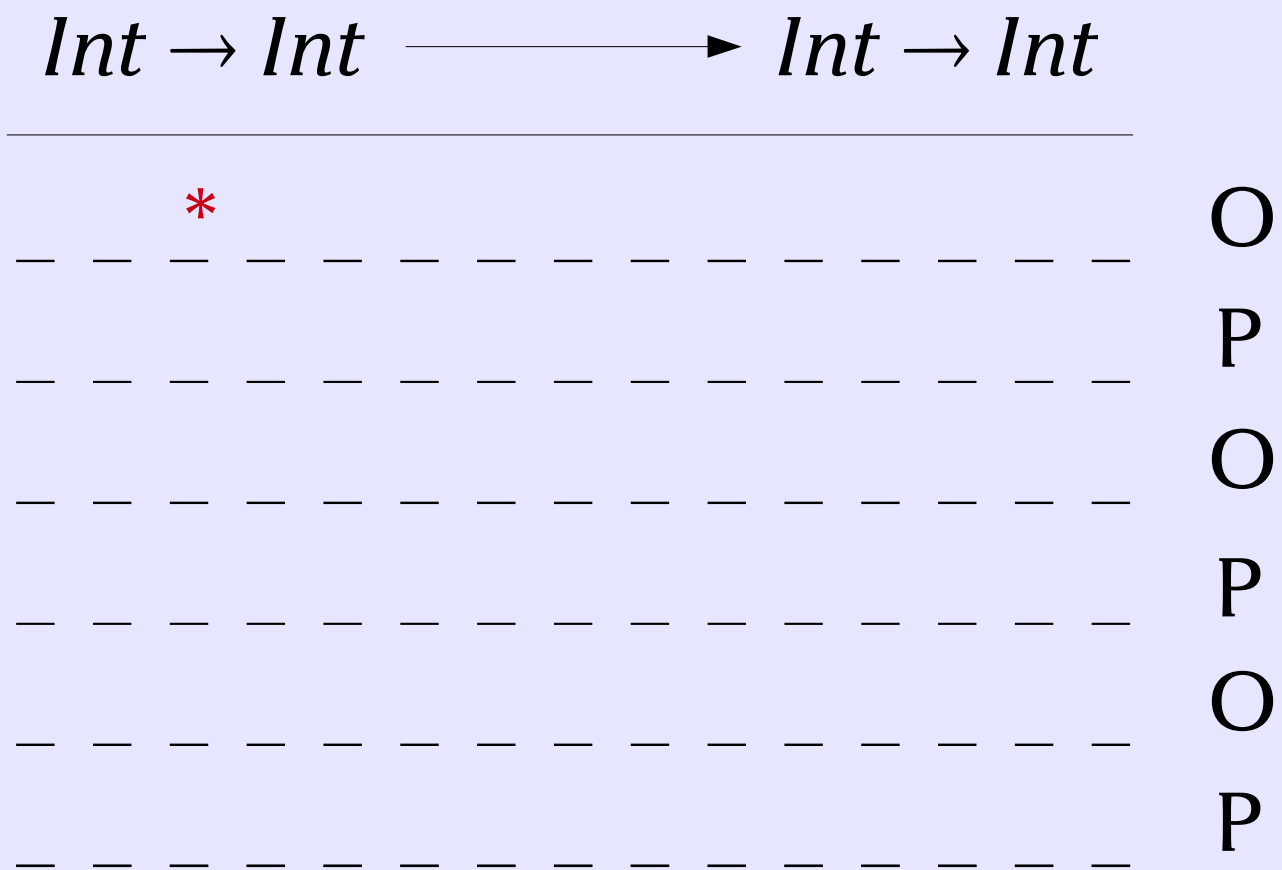
Examples

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



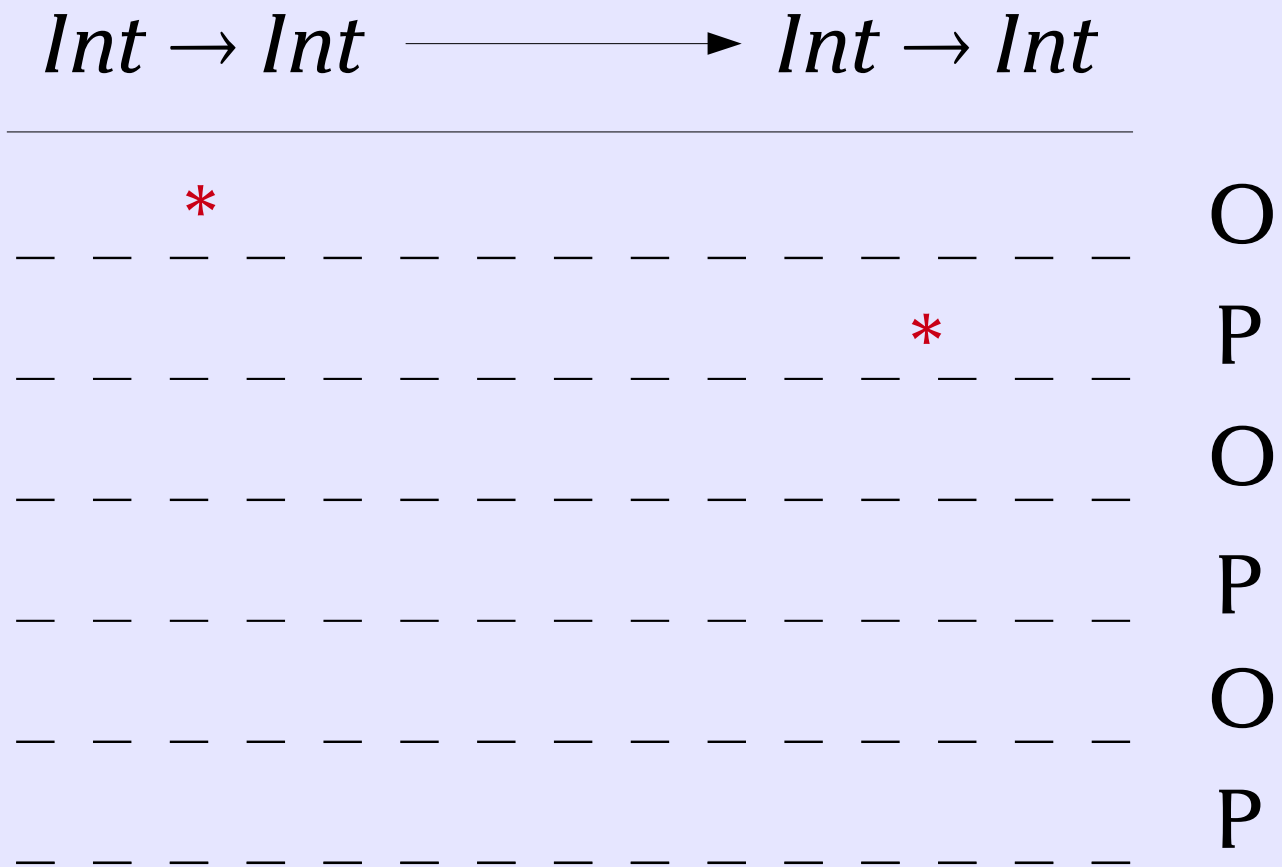
Examples

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



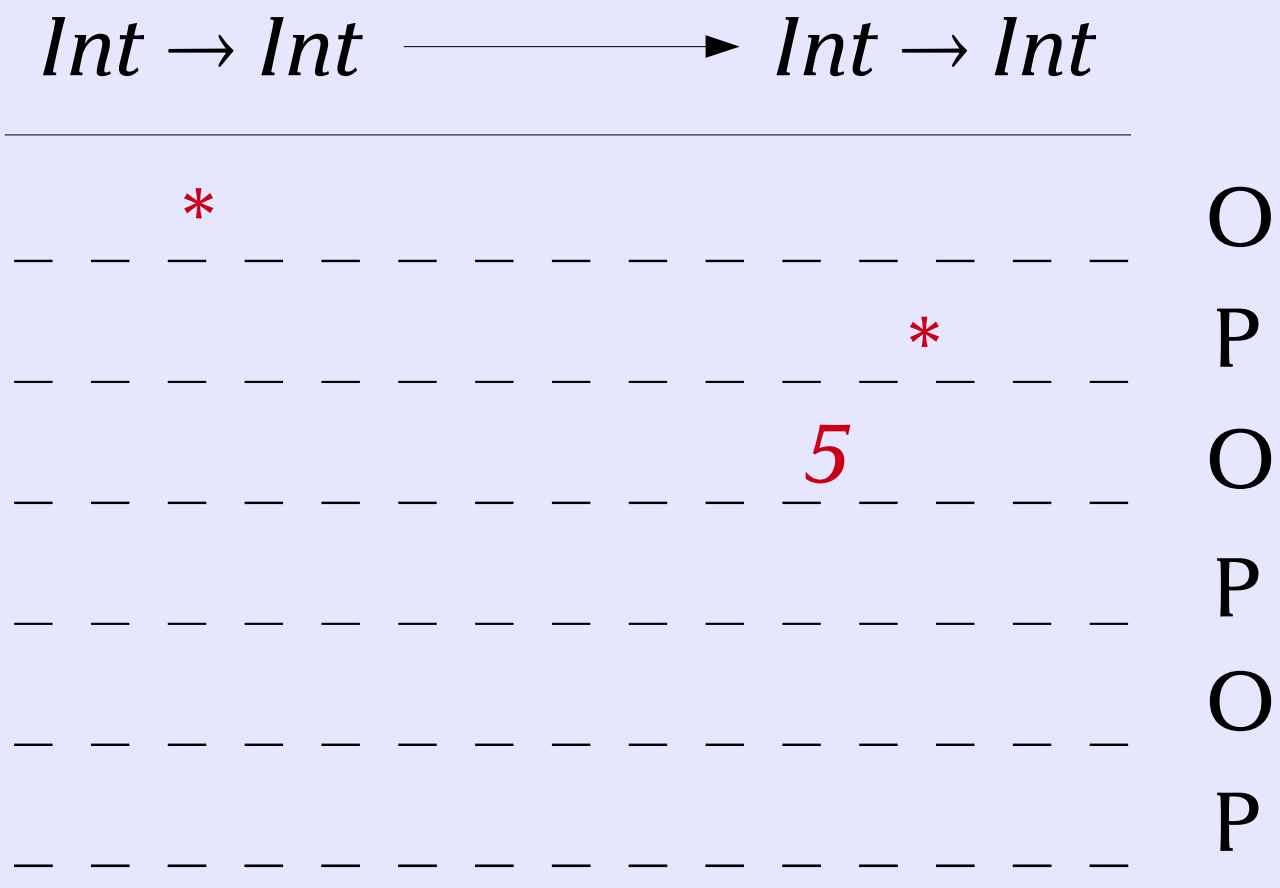
Examples

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



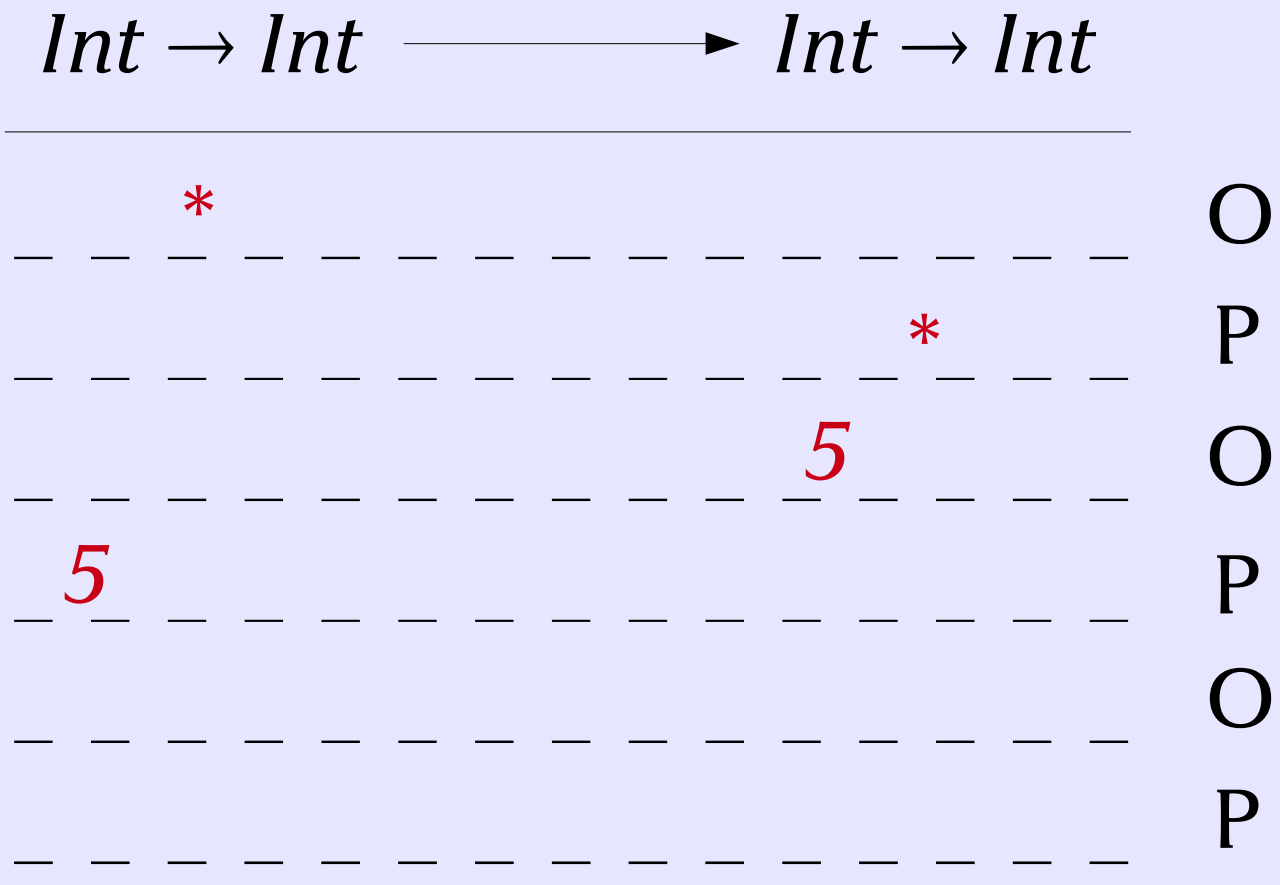
Examples

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



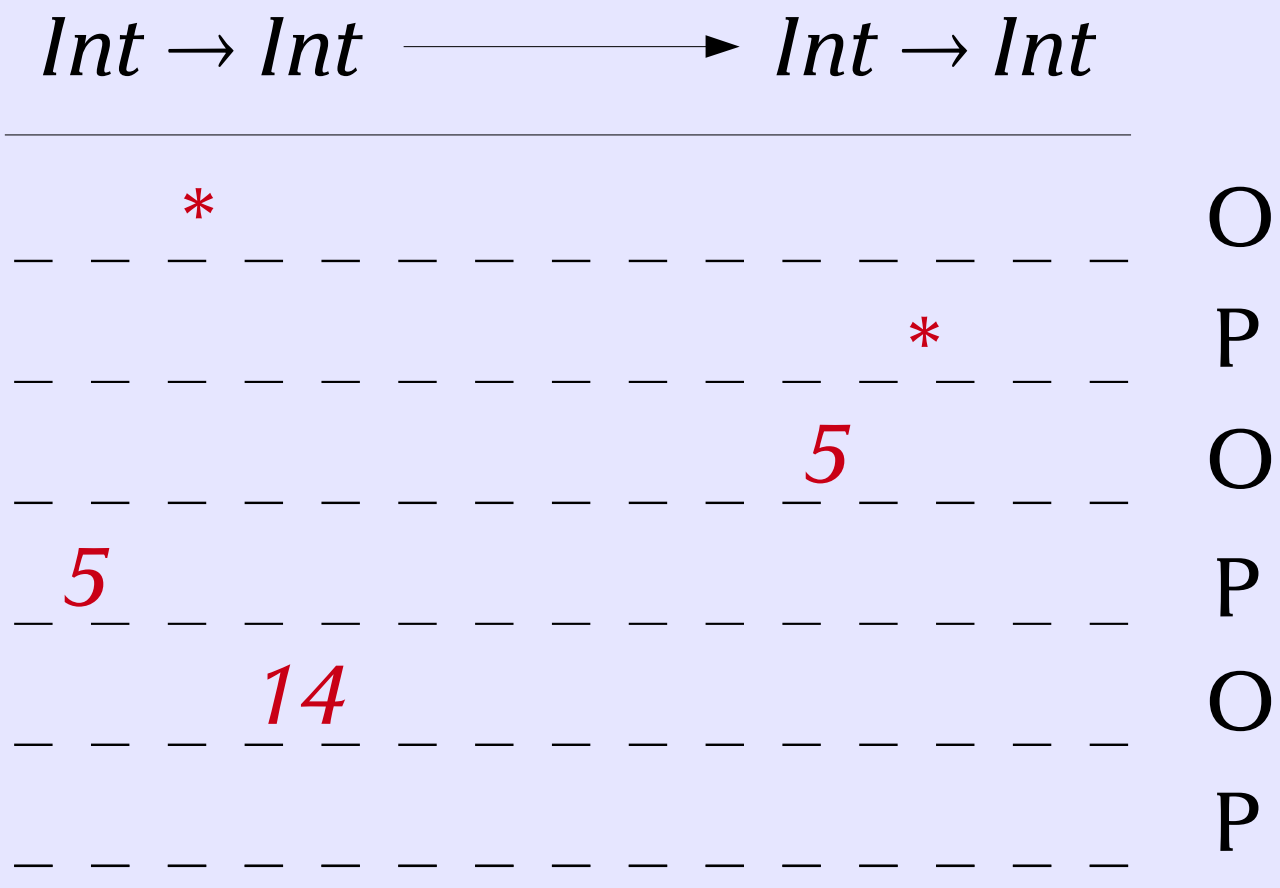
Examples

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



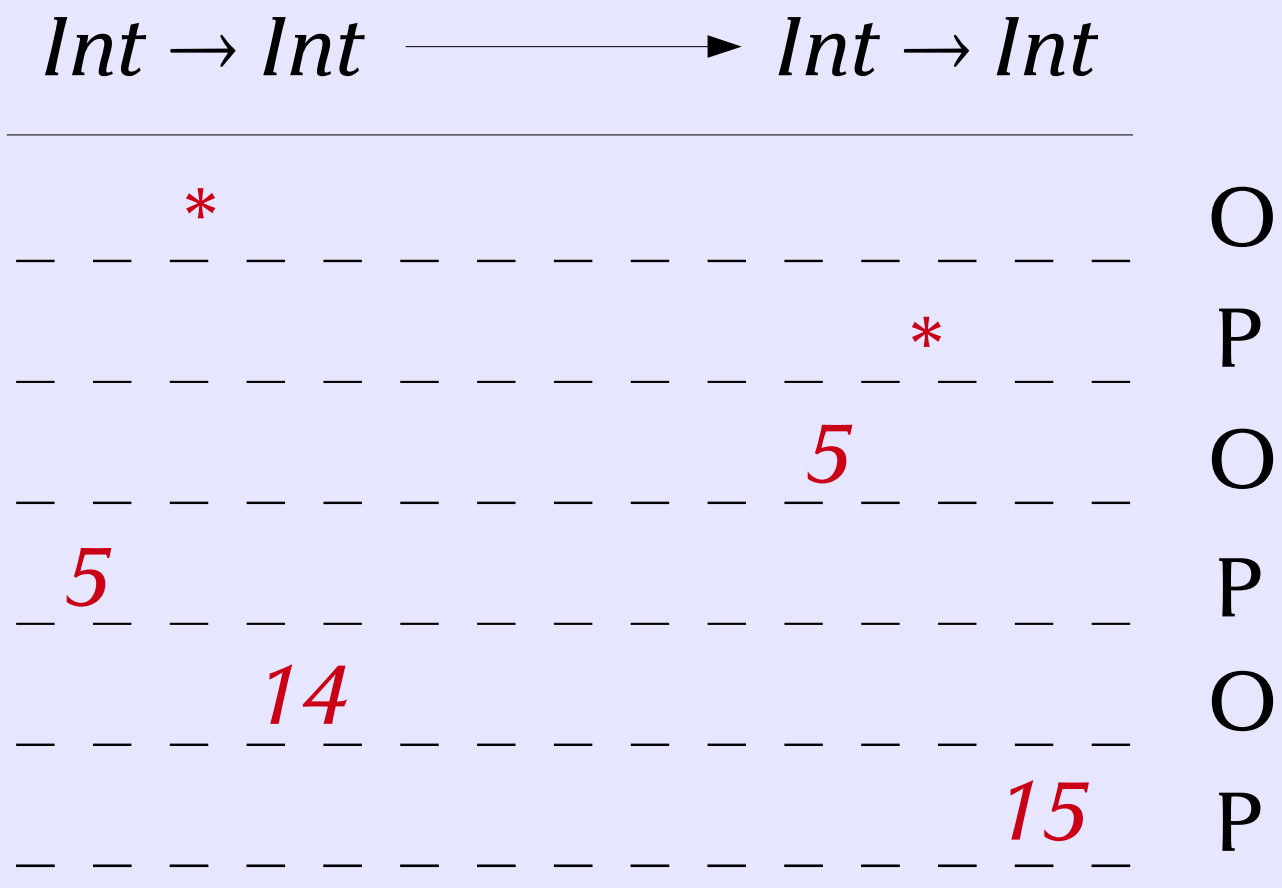
Examples

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



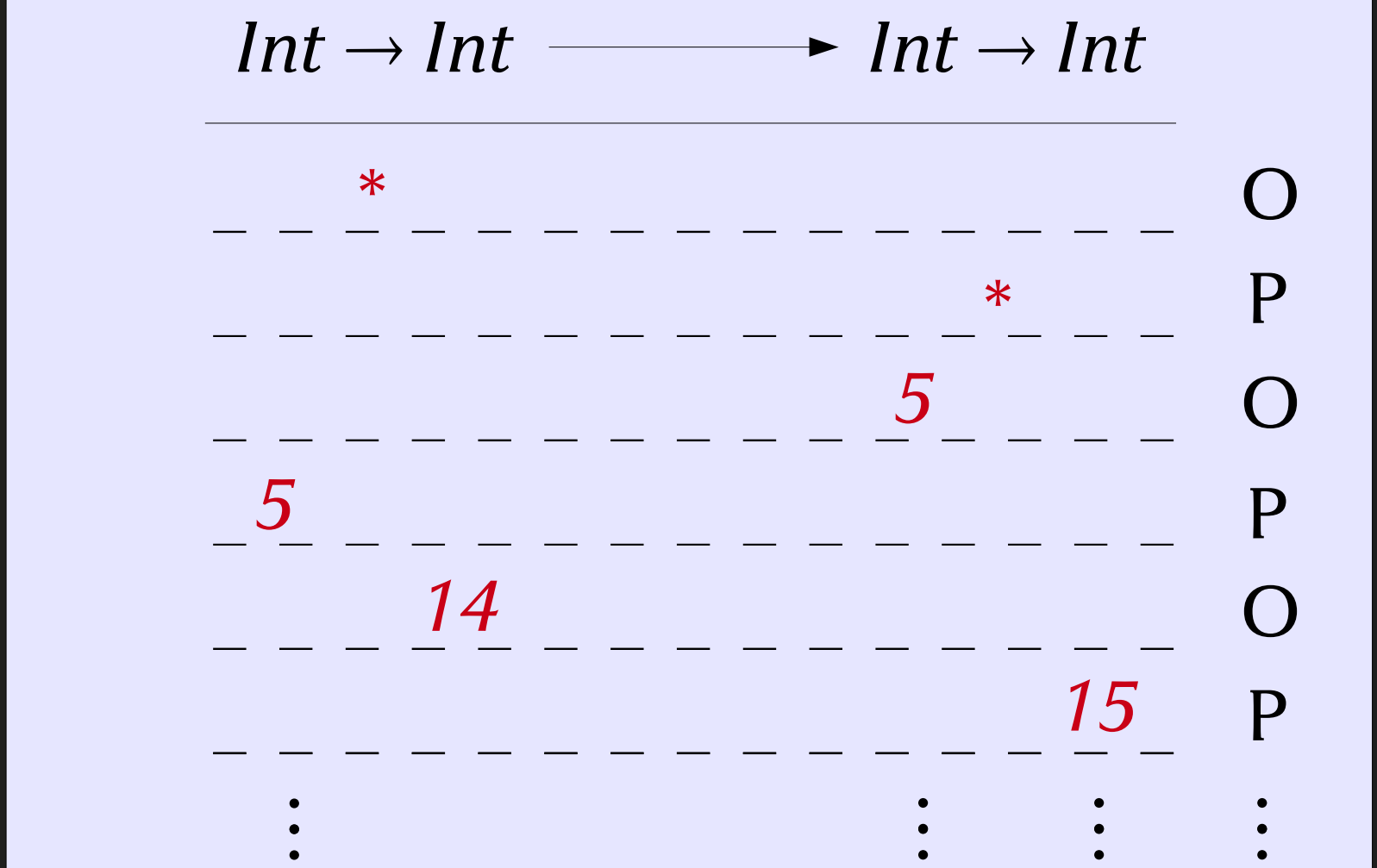
Examples

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



Examples

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



Game Semantics

- Computation is modelled as a 2-player game between:
  - *Opponent* (the environment)
  - *Proponent* (the program)
- Qualitative games

Game Semantics

- Computation is modelled as a 2-player game between:
  - *Opponent* (the environment)
  - *Proponent* (the program)
- Qualitative games
- Programs = *strategies* for Proponent

Game Semantics

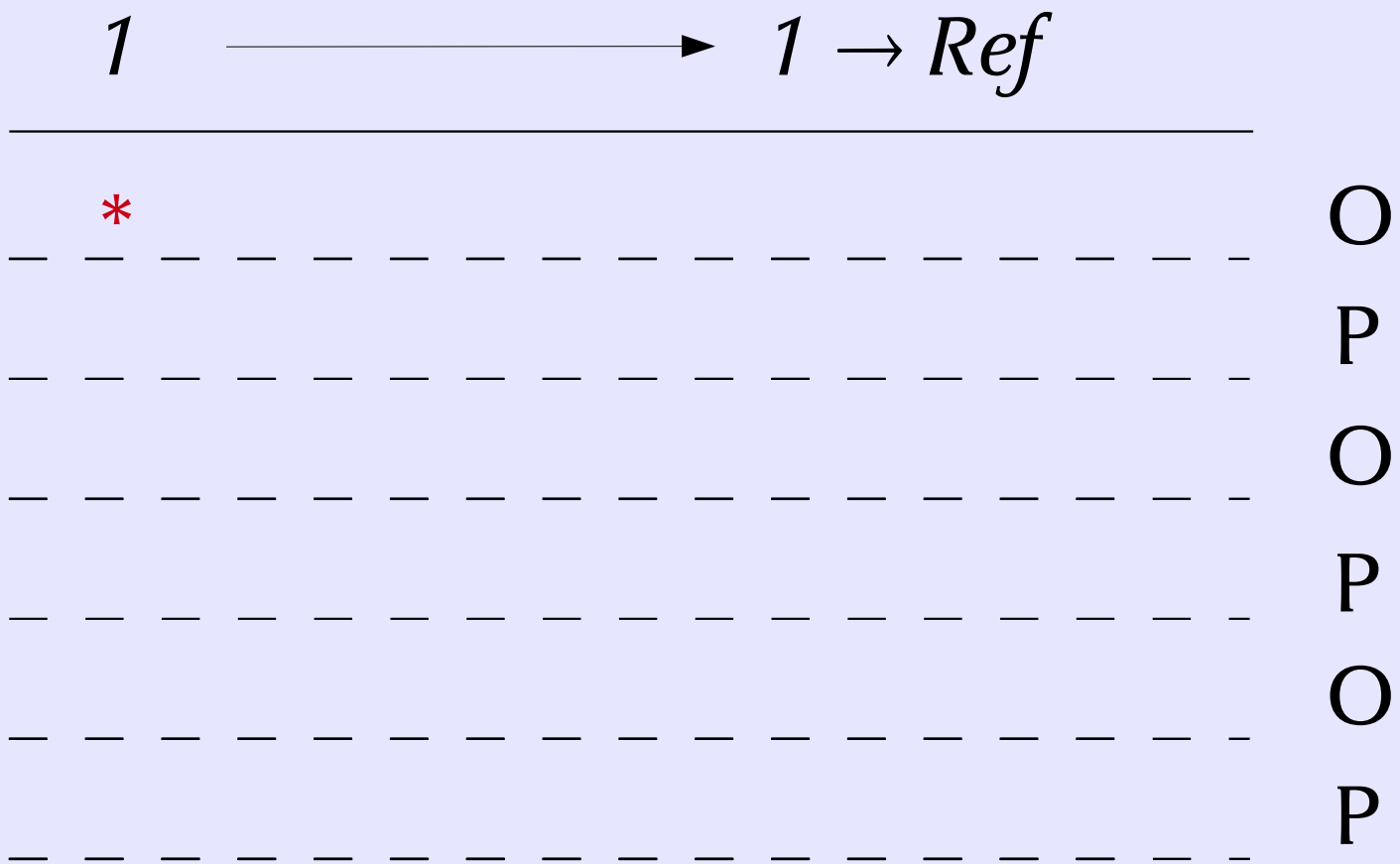
- Computation is modelled as a 2-player game between:
  - *Opponent* (the environment)
  - *Proponent* (the program)
- Qualitative games
- Programs = *strategies* for Proponent
- Families (i.e. *categories*) of games

Games for Reduced ML [*FOSSACS'09*]

- Moves may contain names
- Moves carry store

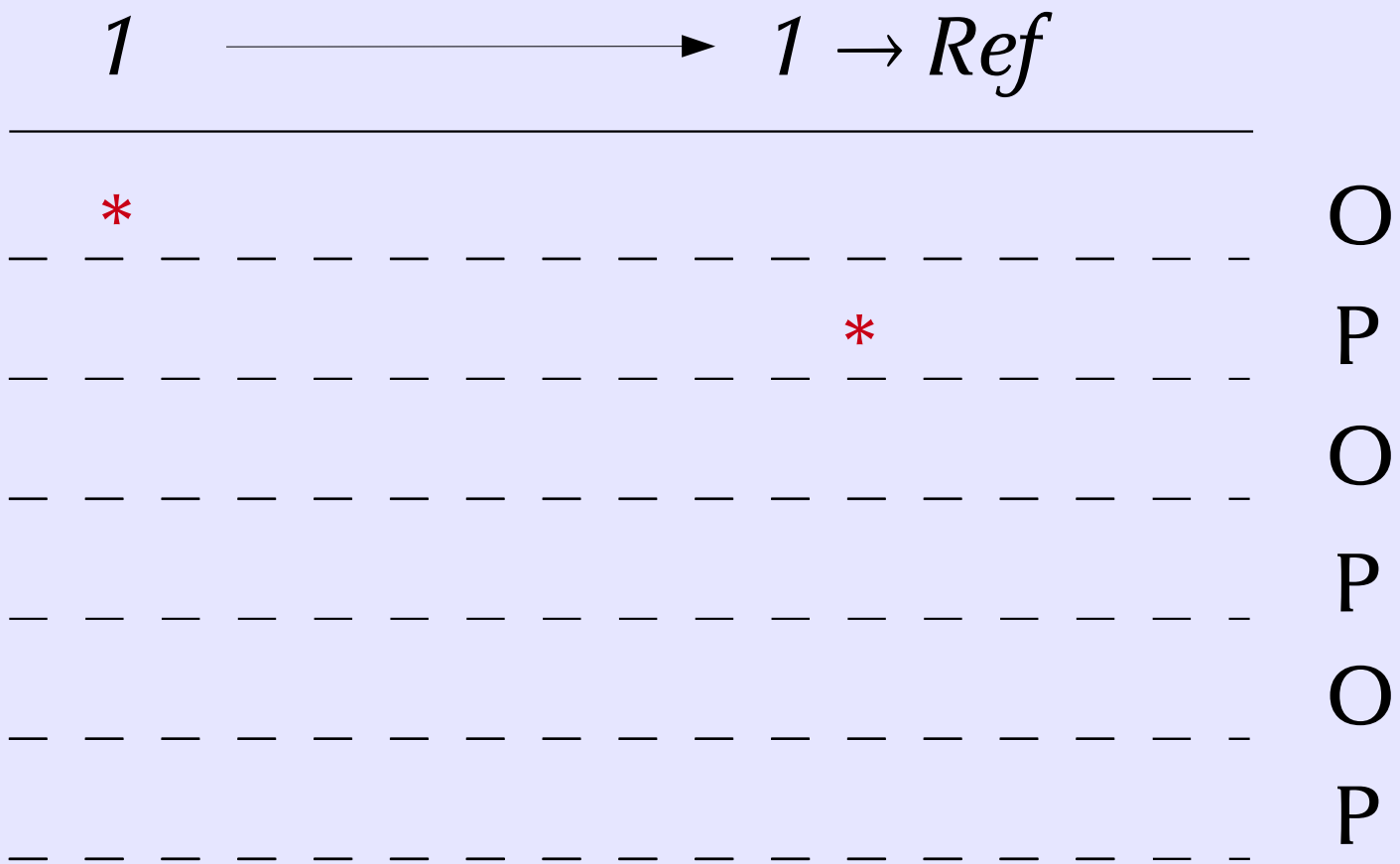
Examples

$\vdash \lambda z.\mathtt{ref}(0) : \mathtt{unit} \to \mathtt{intref}$



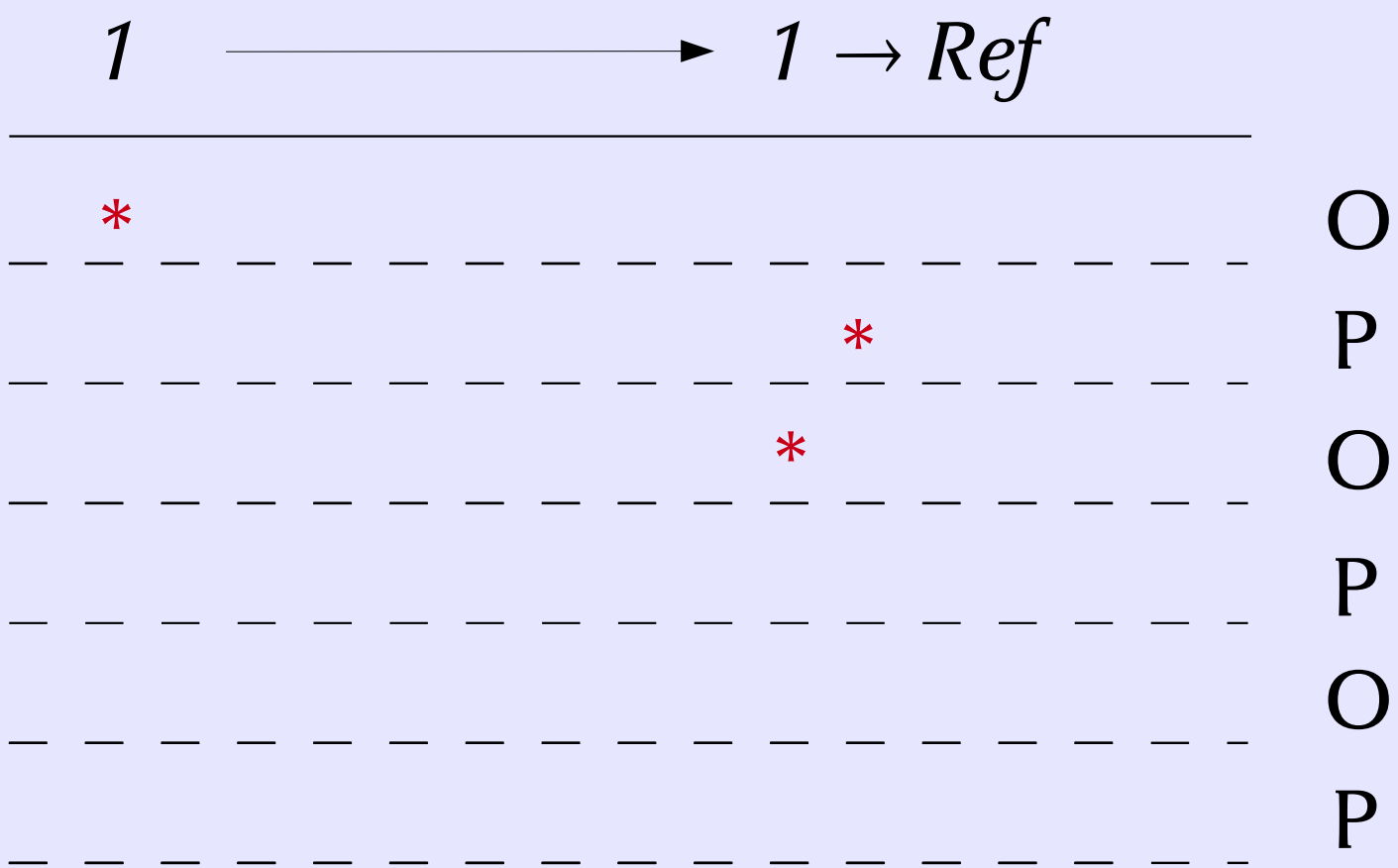
Examples

$\vdash \lambda z.\mathtt{ref}(0) : \mathtt{unit} \to \mathtt{intref}$



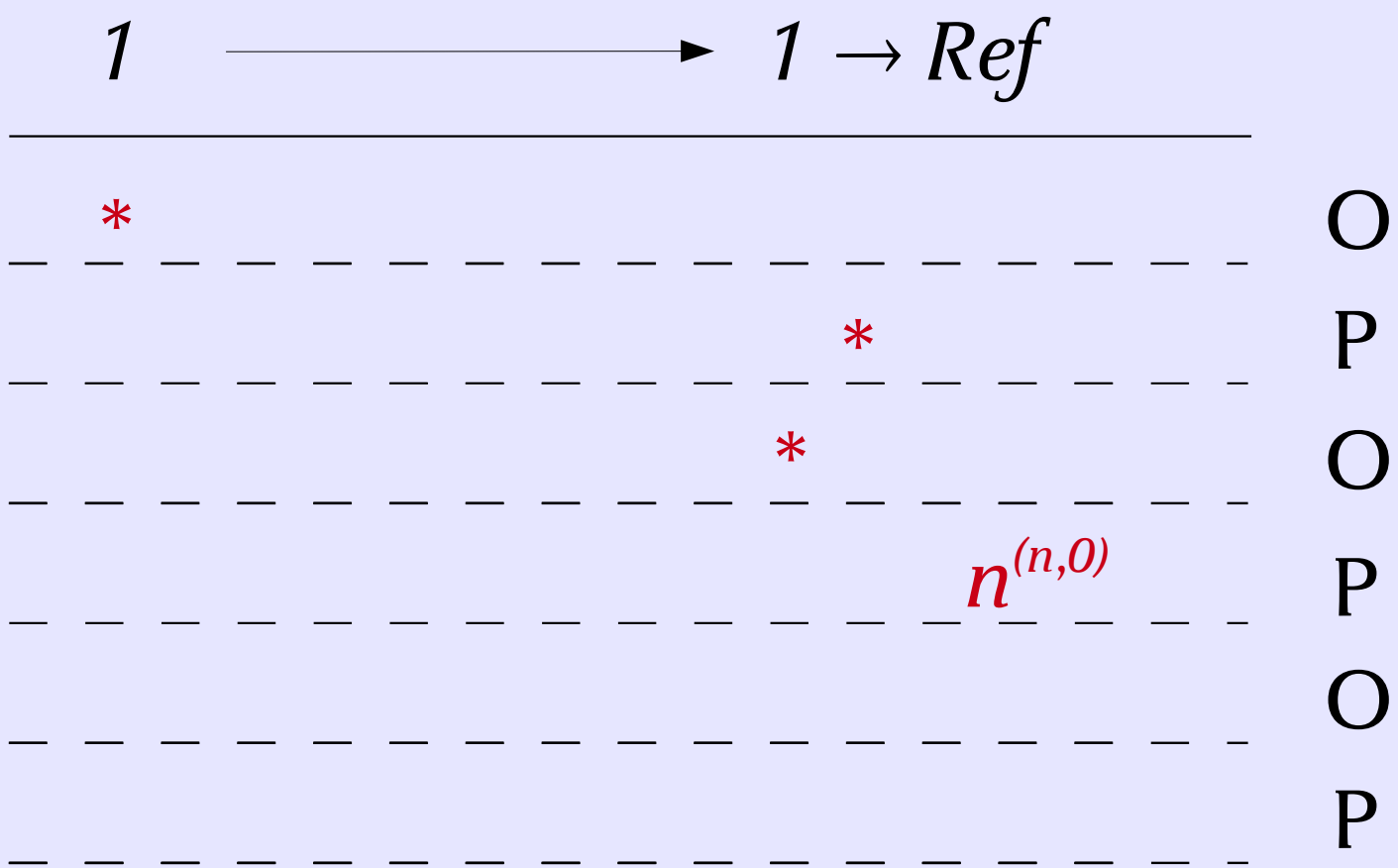
Examples

$\vdash \lambda z.\mathtt{ref}(0) : \mathtt{unit} \to \mathtt{intref}$



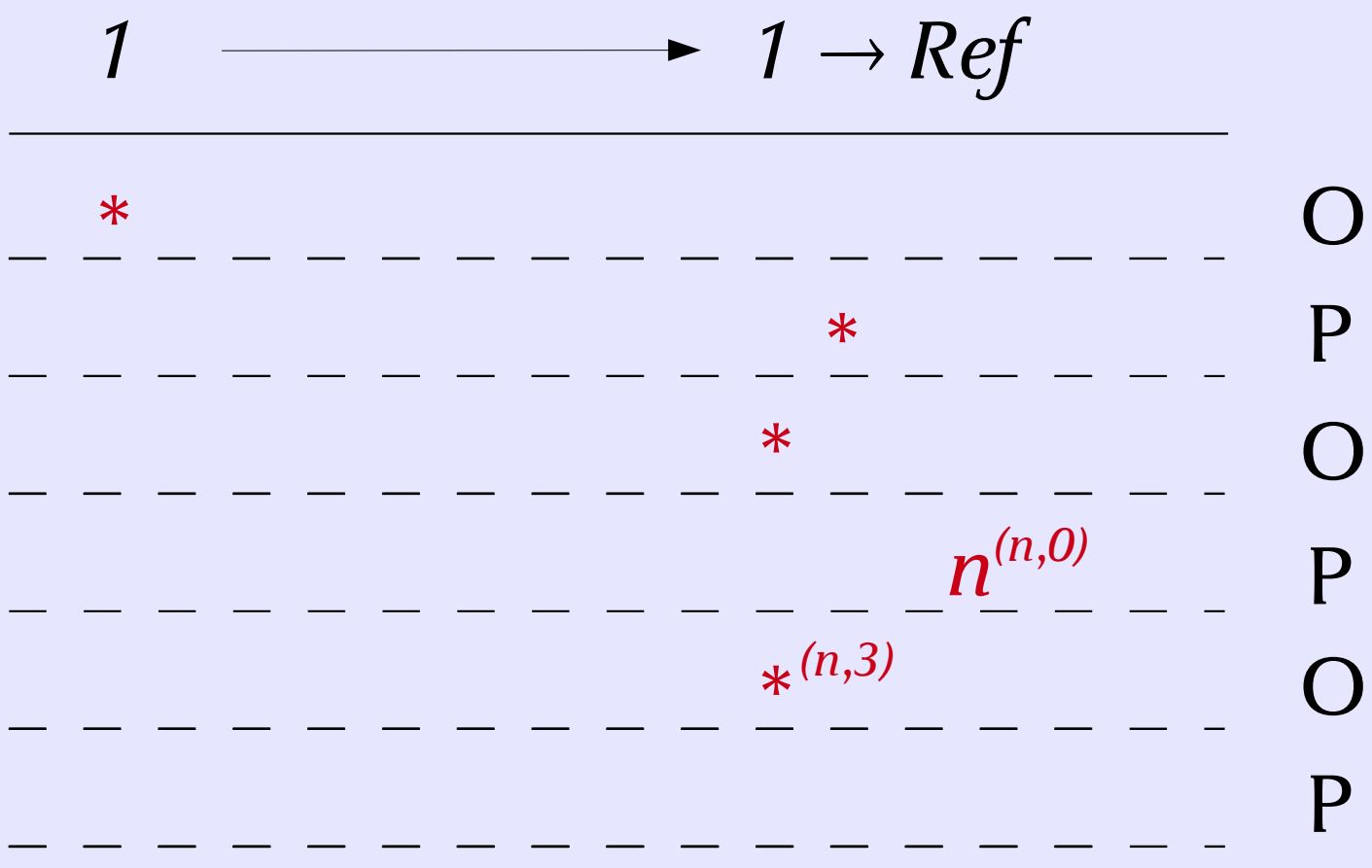
Examples

$\vdash \lambda z.\mathtt{ref}(0) : \mathtt{unit} \to \mathtt{intref}$



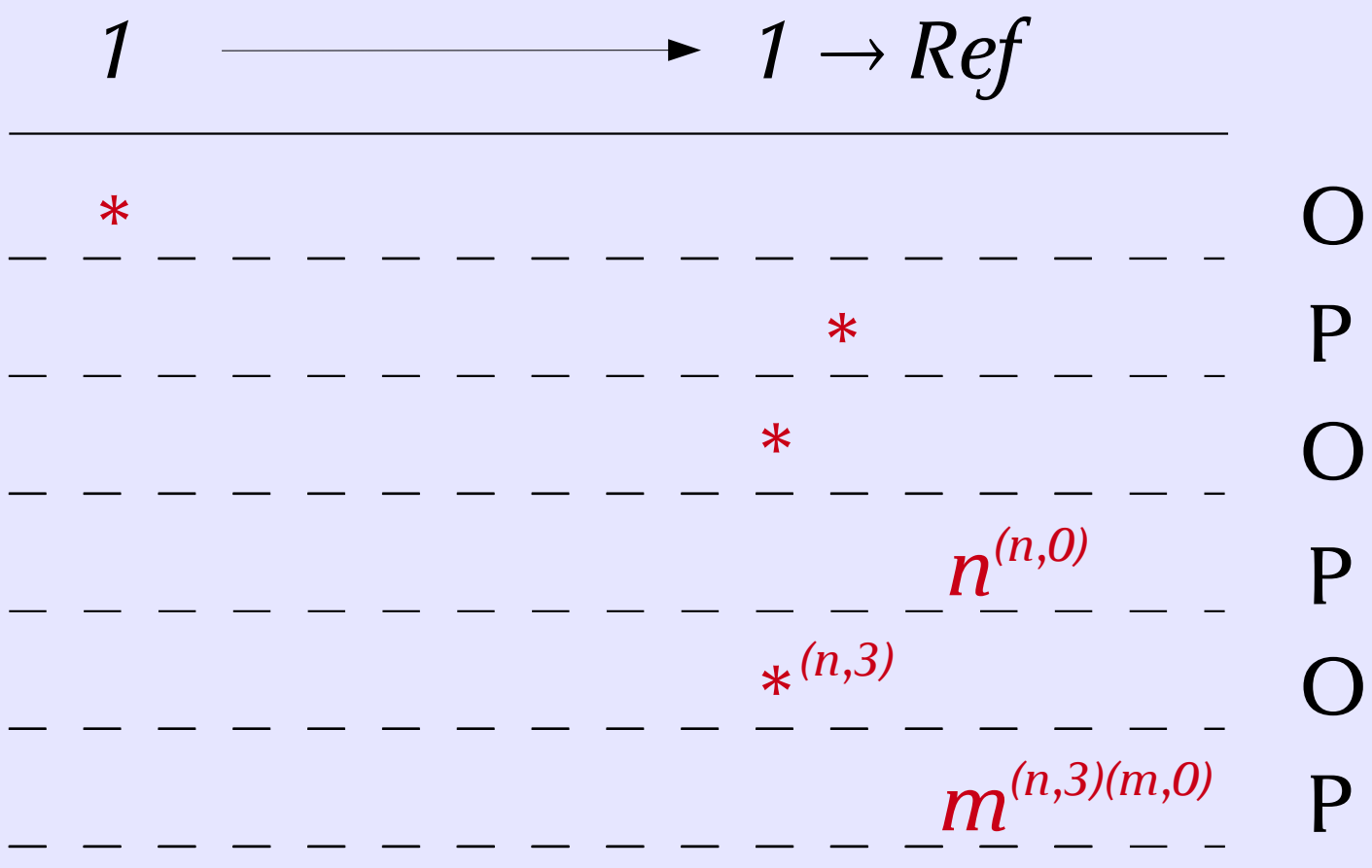
Examples

$\vdash \lambda z.\mathtt{ref}(0) : \mathtt{unit} \to \mathtt{intref}$



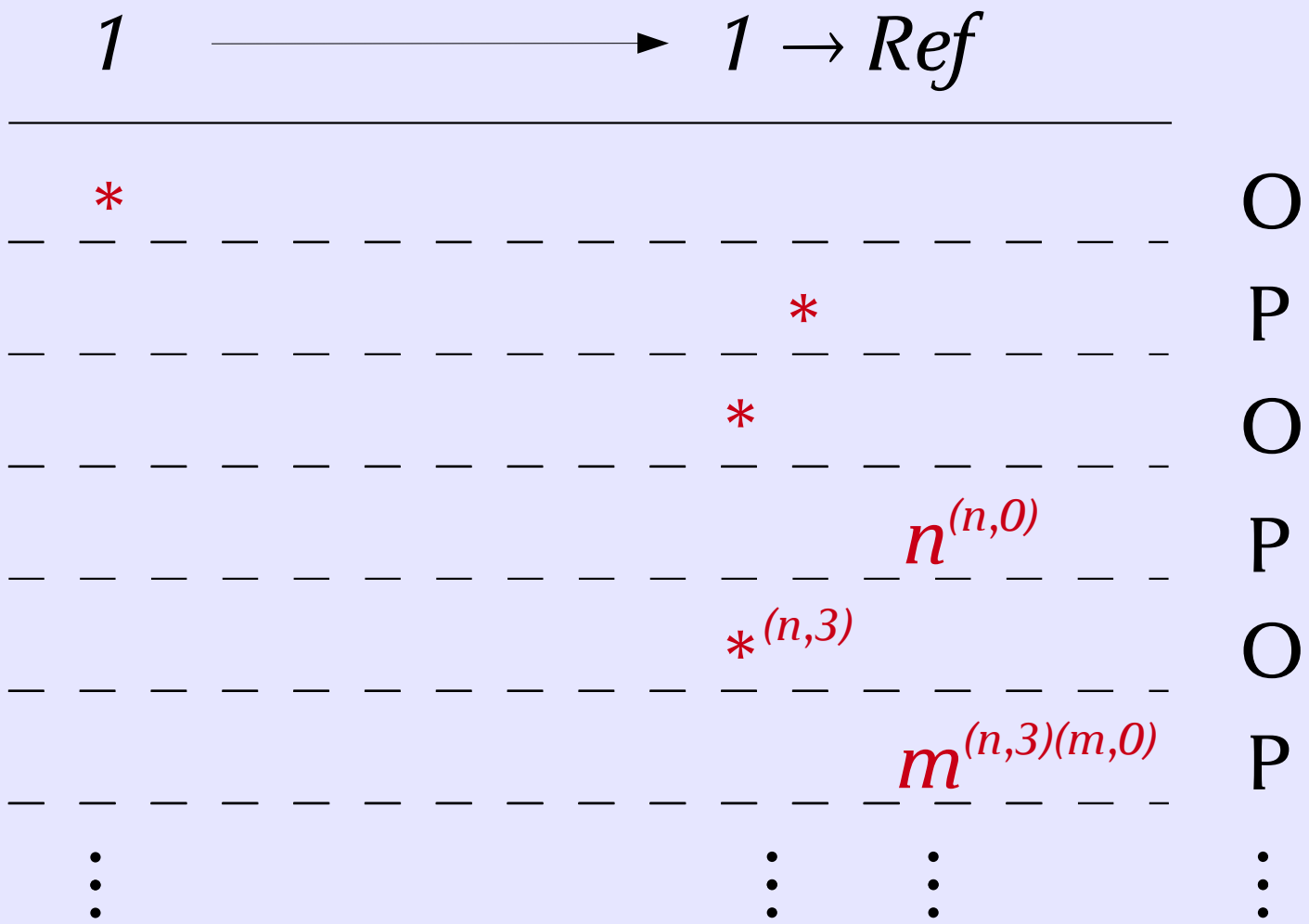
Examples

$\vdash \lambda z.\mathtt{ref}(0) : \mathtt{unit} \to \mathtt{intref}$



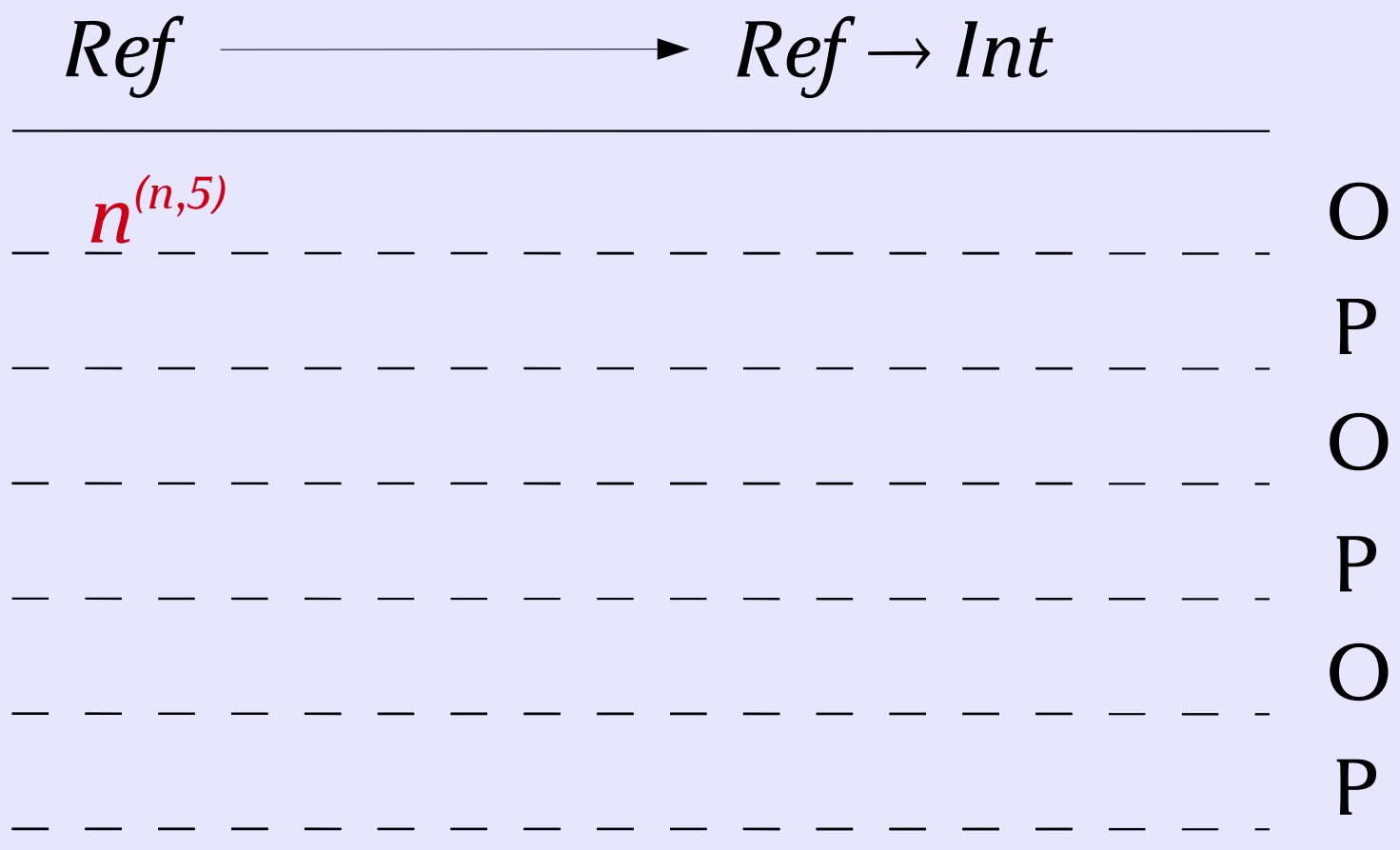
Examples

$\vdash \lambda z.\mathtt{ref}(0) : \mathtt{unit} \to \mathtt{intref}$



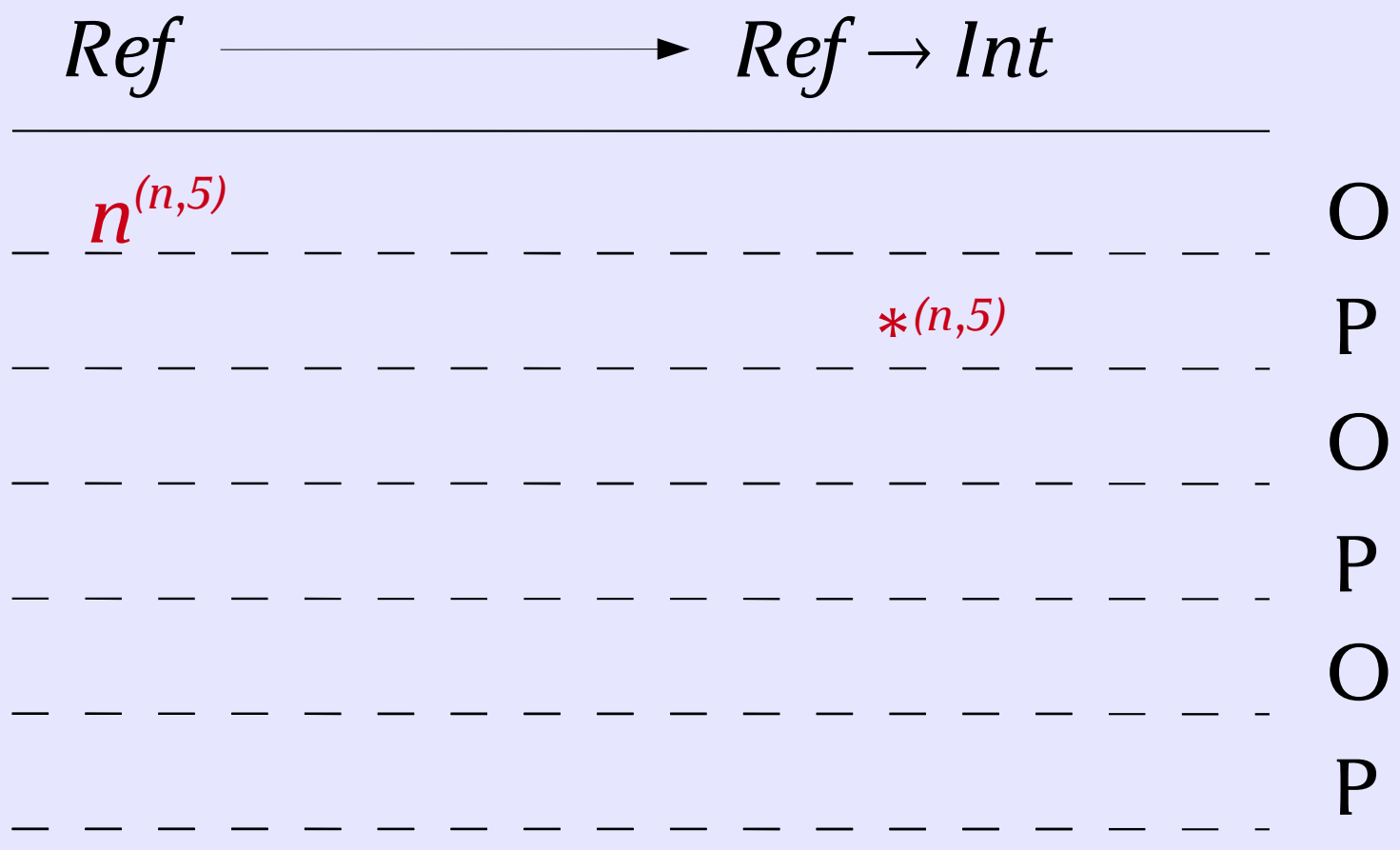
Examples

$\texttt{x : intref} \vdash \lambda\texttt{y.(x == y) : intref} \to \texttt{int}$



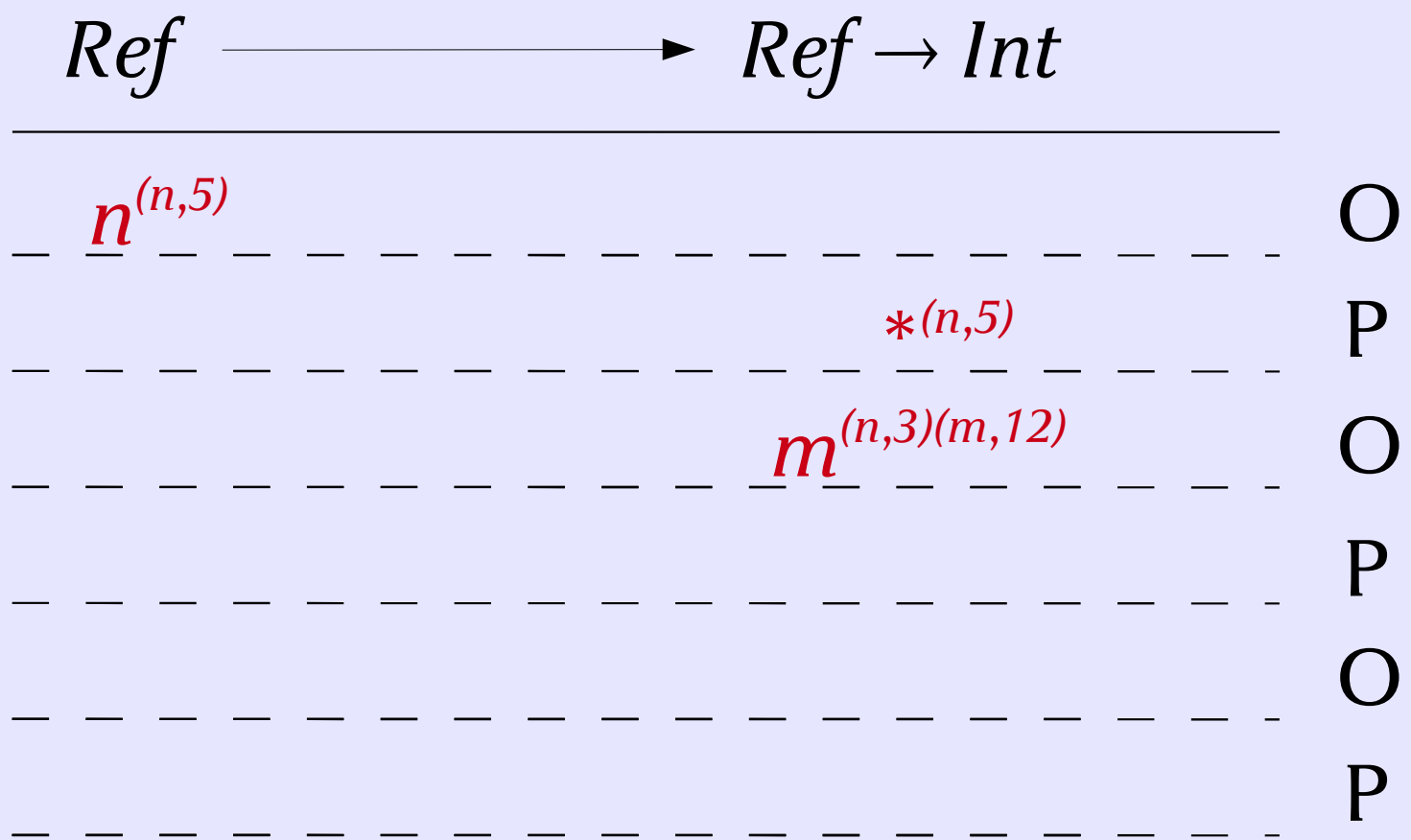
Examples

$\texttt{x : intref} \vdash \lambda\texttt{y.(x == y) : intref} \to \texttt{int}$



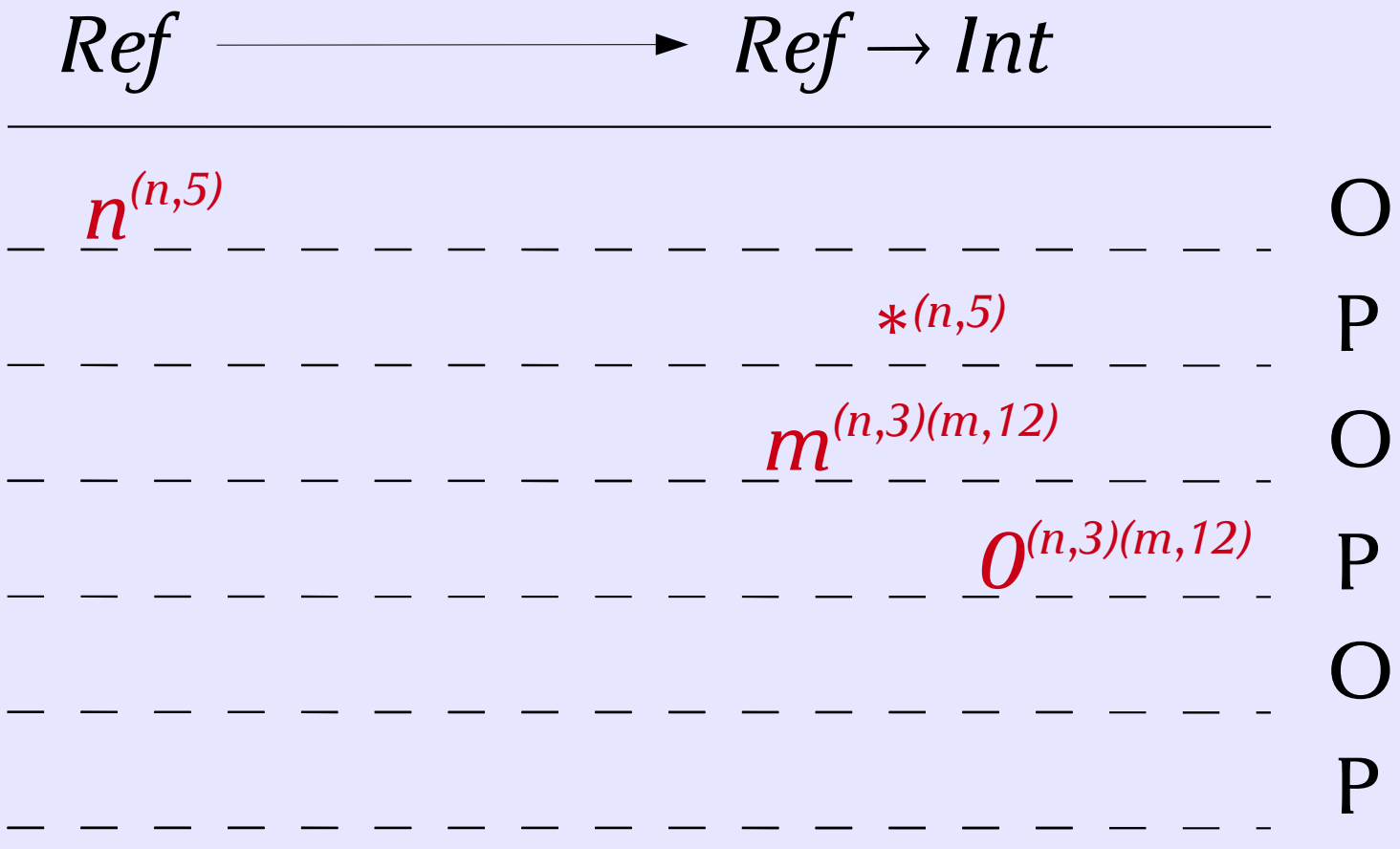
Examples

$\texttt{x : intref} \vdash \lambda\texttt{y.(x == y) : intref} \to \texttt{int}$



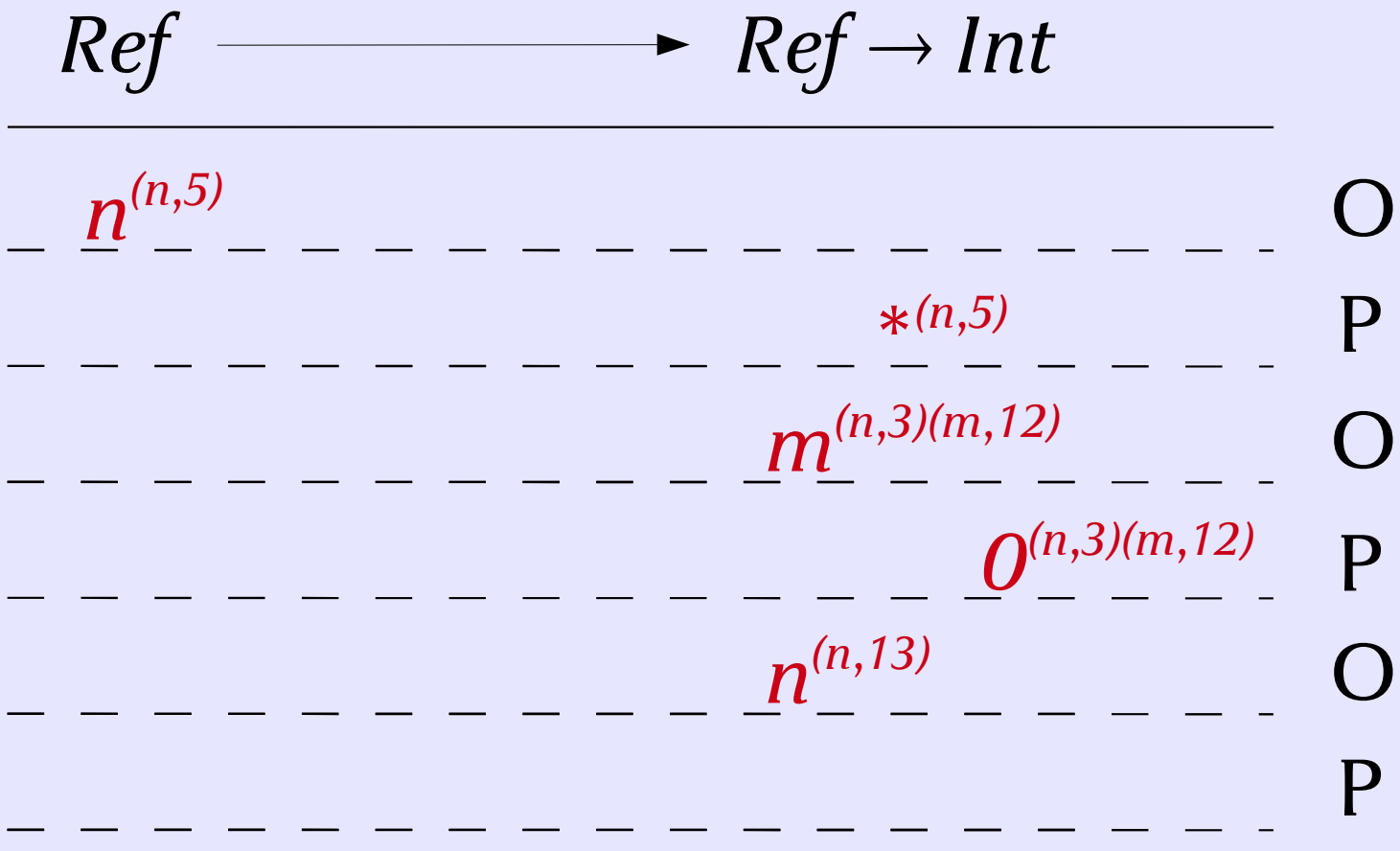
Examples

$\texttt{x : intref} \vdash \lambda\texttt{y.(x == y) : intref} \to \texttt{int}$



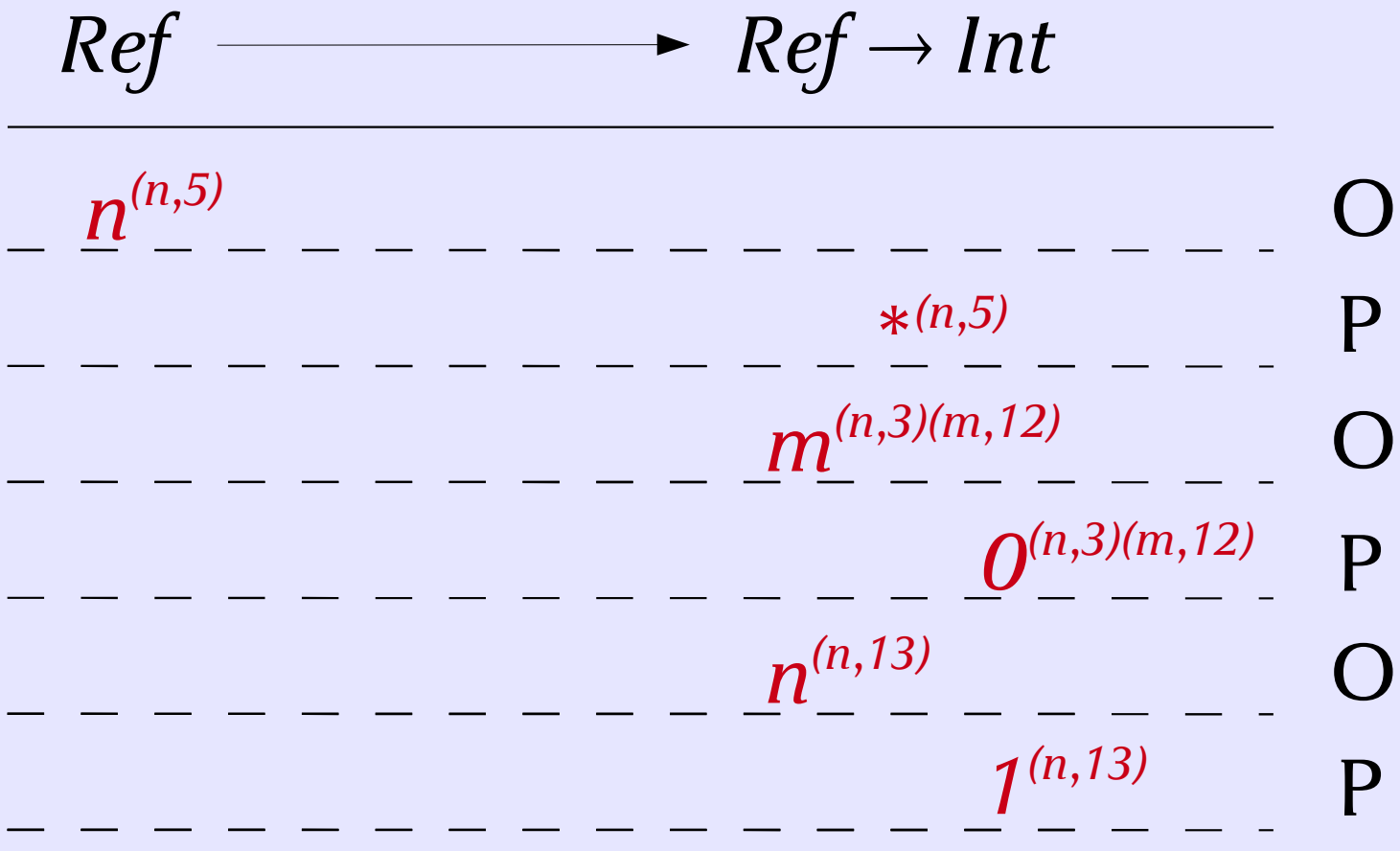
Examples

$\texttt{x : intref} \vdash \lambda\texttt{y.(x == y) : intref} \to \texttt{int}$



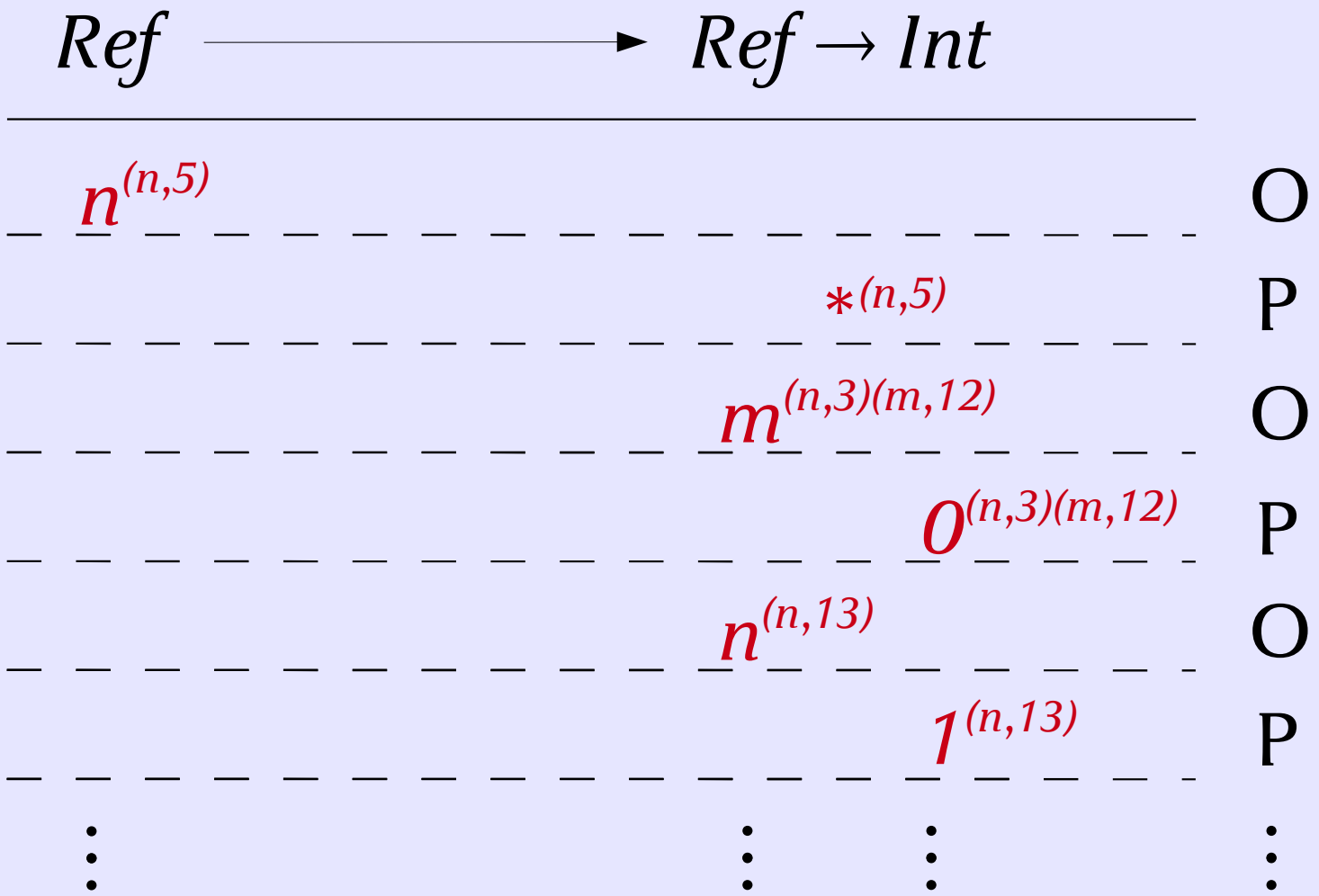
Examples

$\texttt{x : intref} \vdash \lambda\texttt{y.(x == y) : intref} \to \texttt{int}$



Examples

$\texttt{x : intref} \vdash \lambda\texttt{y.(x == y) : intref} \to \texttt{int}$



Games for Reduced ML [*FOSSACS'09*]

- Moves may contain names
- Moves carry store
- 1st construction: $M \cong N \iff [\![M]\!] \cong [\![N]\!]$

Games for Reduced ML [*FOSSACS'09*]

- Moves may contain names
- Moves carry store
- 1st construction: $M \cong N \Leftrightarrow [\![M]\!] \cong [\![N]\!]$
- 2nd construction: $M \cong N \Leftrightarrow \overline{[\![M]\!]} = \overline{[\![N]\!]}$

mini-RedML

mini-RedML

- Restrict integers to: $\{ 0, 1, \ldots, i_{\max} \}$

mini-RedML

- Restrict integers to: $\{ 0, 1, \ldots, i_{\text{max}} \}$
- Restrict types to:

$$\theta \texttt{ ::= } \beta \texttt{ | } \beta \to \beta$$

$$\beta \texttt{ ::= unit | int | intref}$$

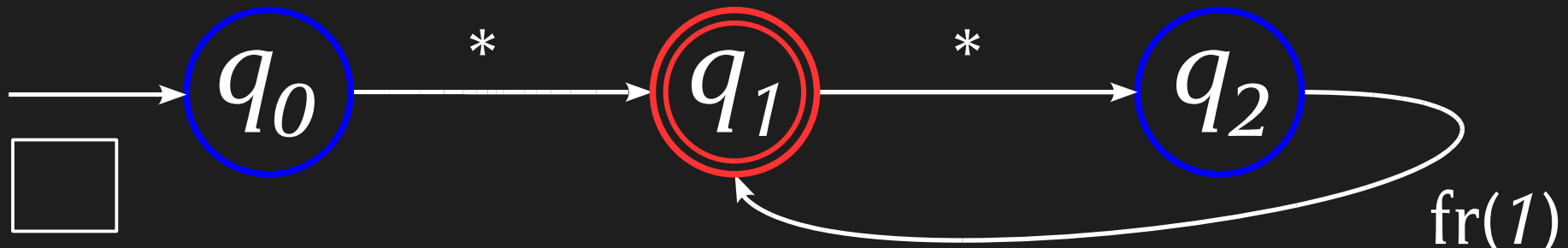
Fresh-register automata [*POPL'11*]

- Infinite alphabet $\mathcal{N}$
- *Freshness* recognition

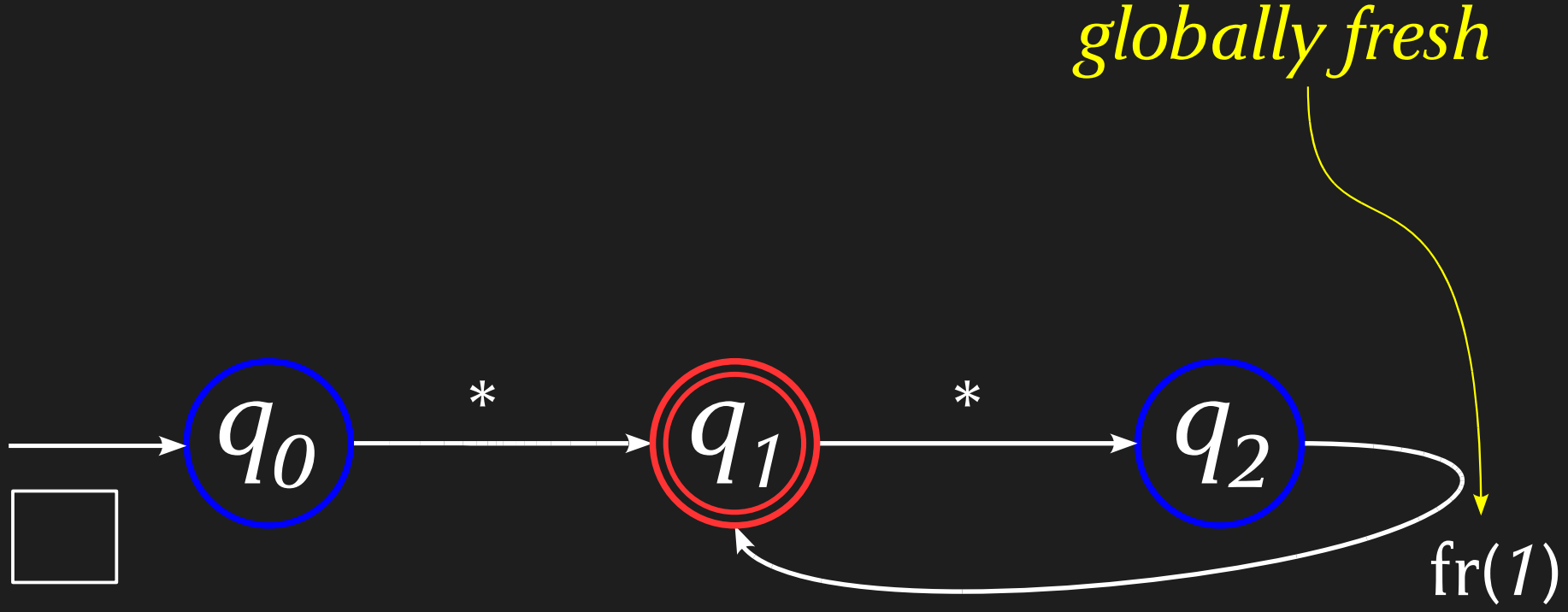
Fresh-register automata [*POPL'11*]

- Finite-state machines with registers
- Infinite alphabet $\mathcal{N}$
- *Freshness* recognition

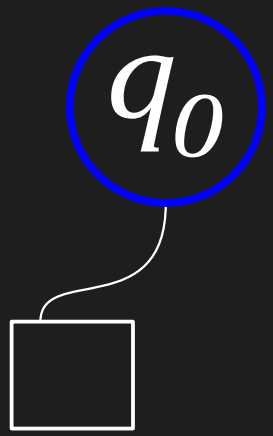
Fresh-register automata

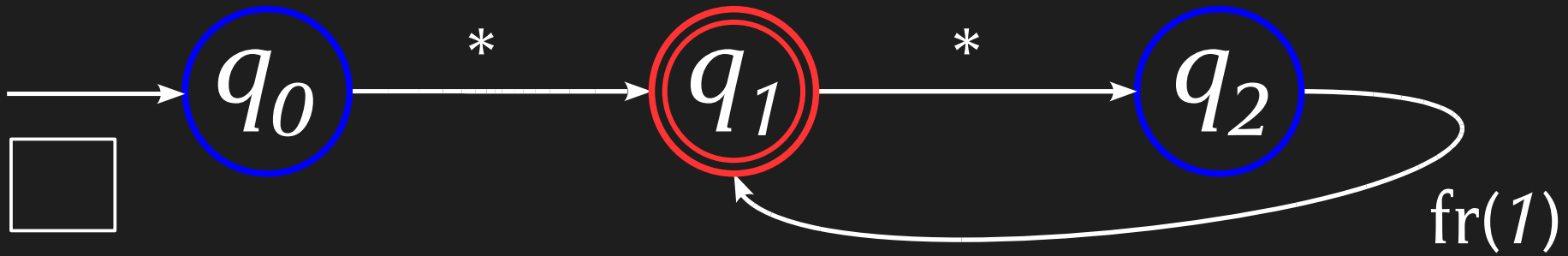


Fresh-register automata

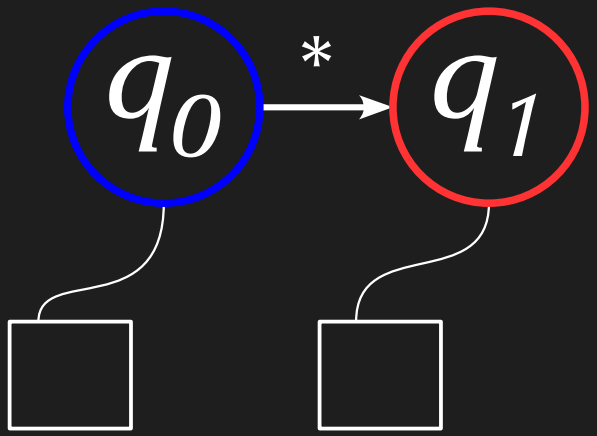


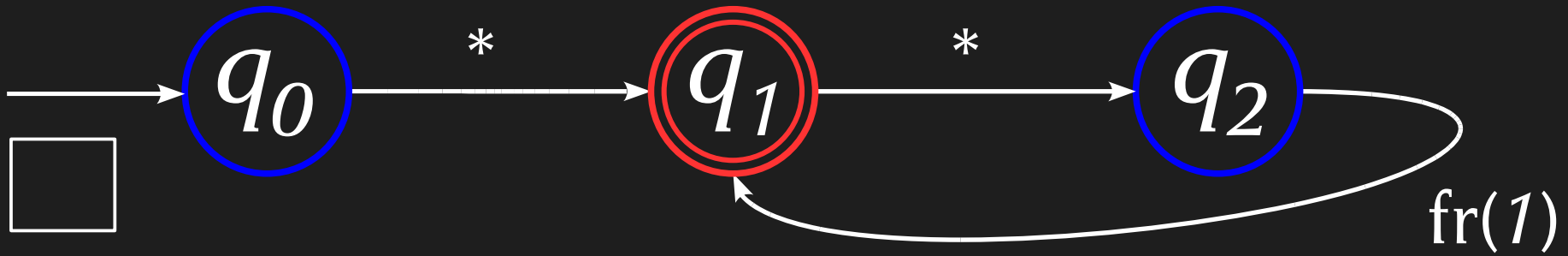
Fresh-register automata



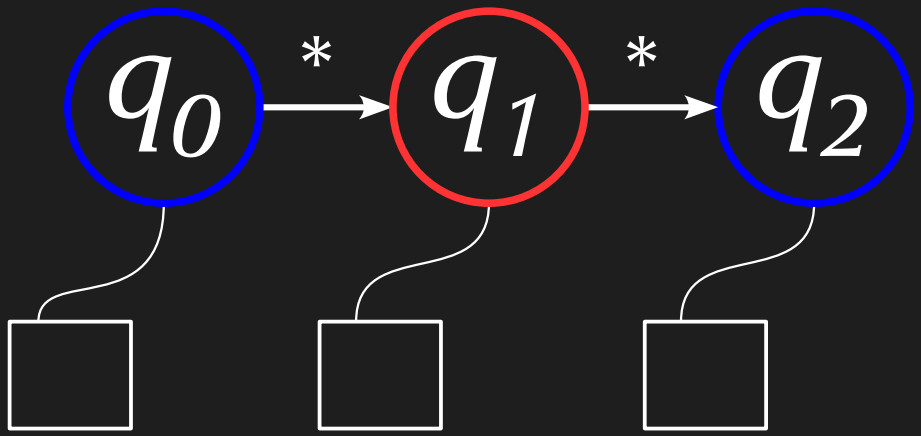


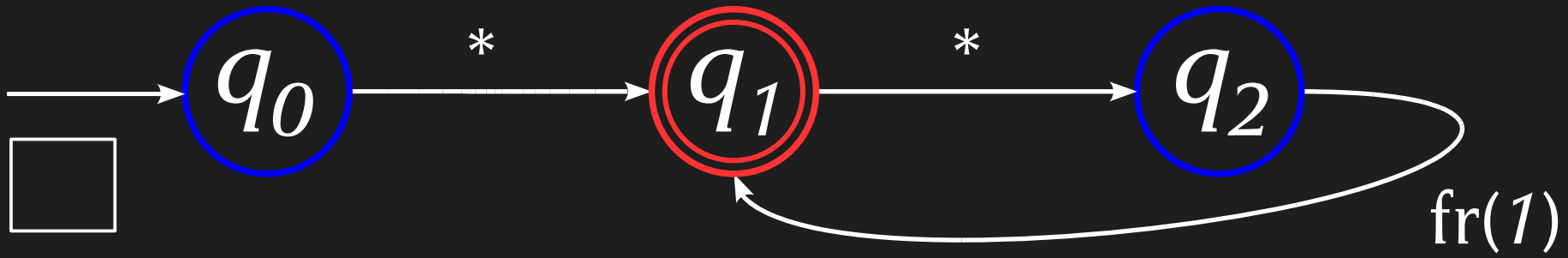
Fresh-register automata



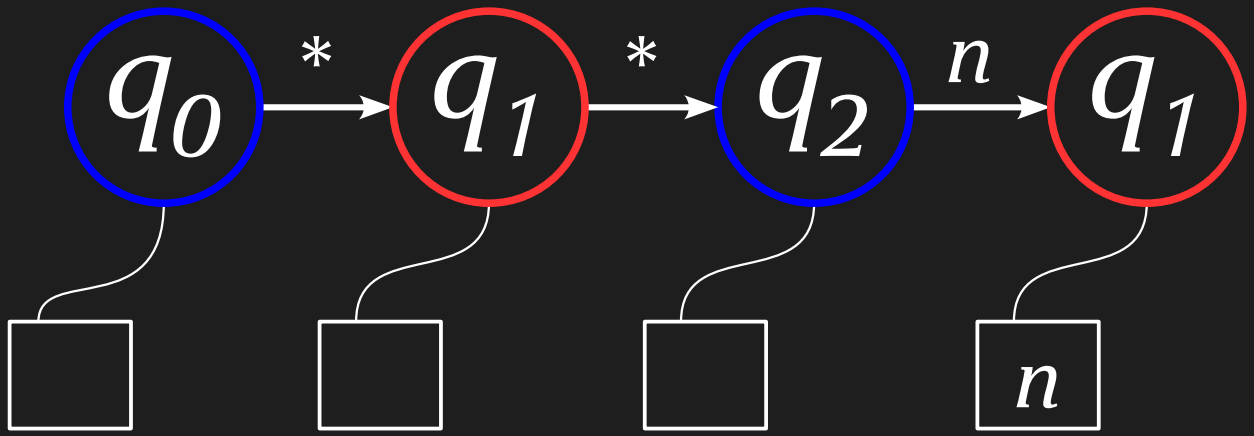


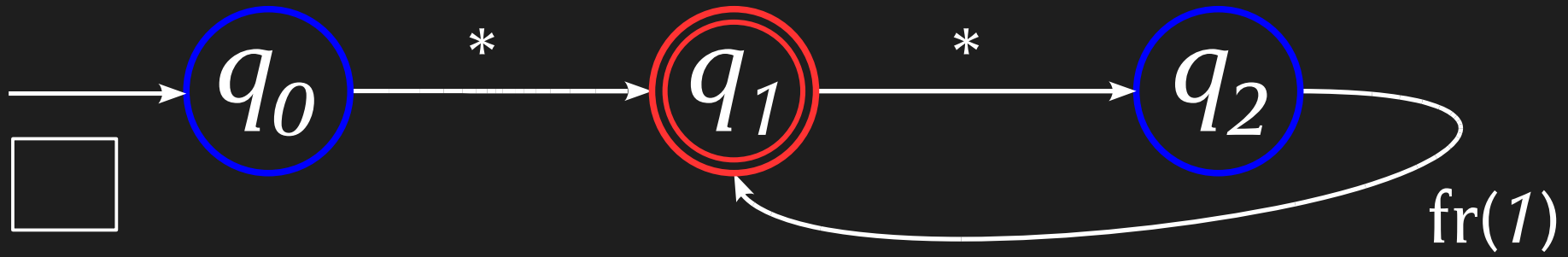
Fresh-register automata



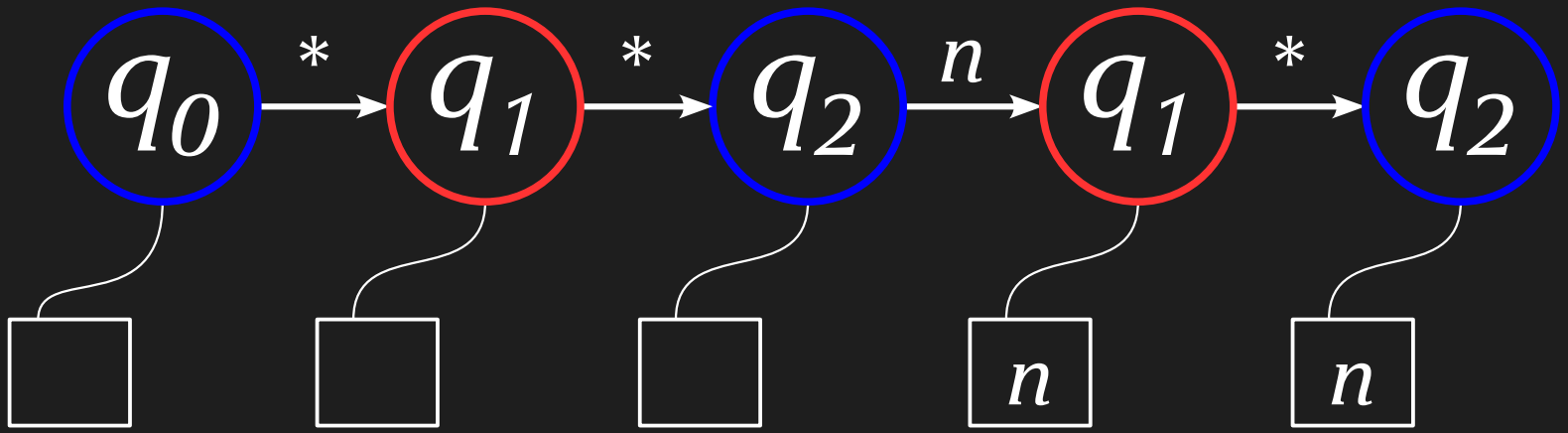


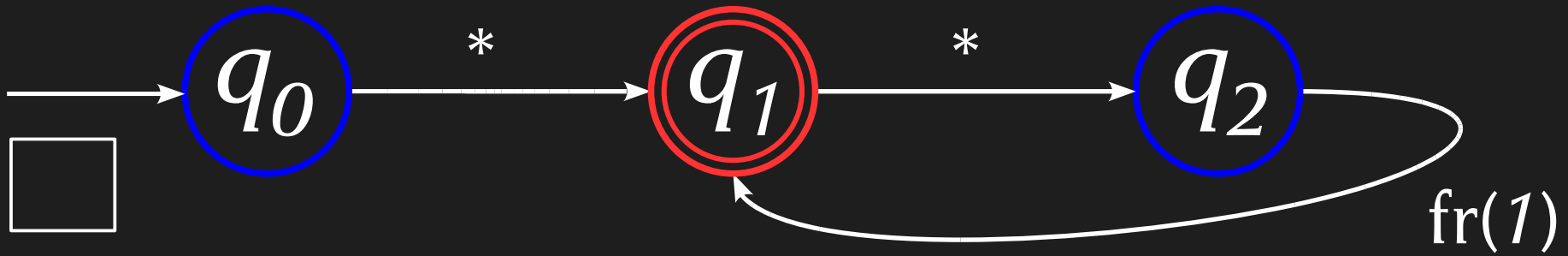
Fresh-register automata



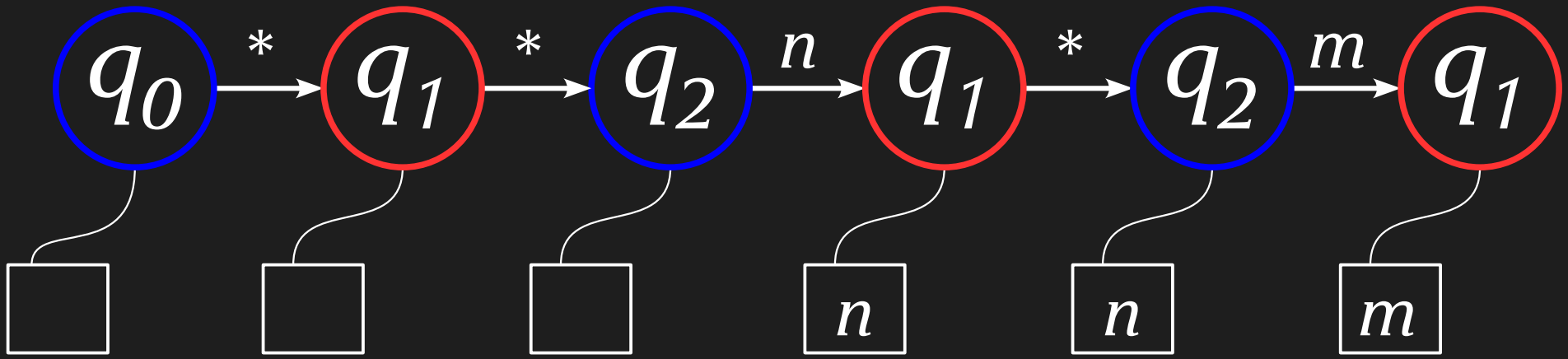


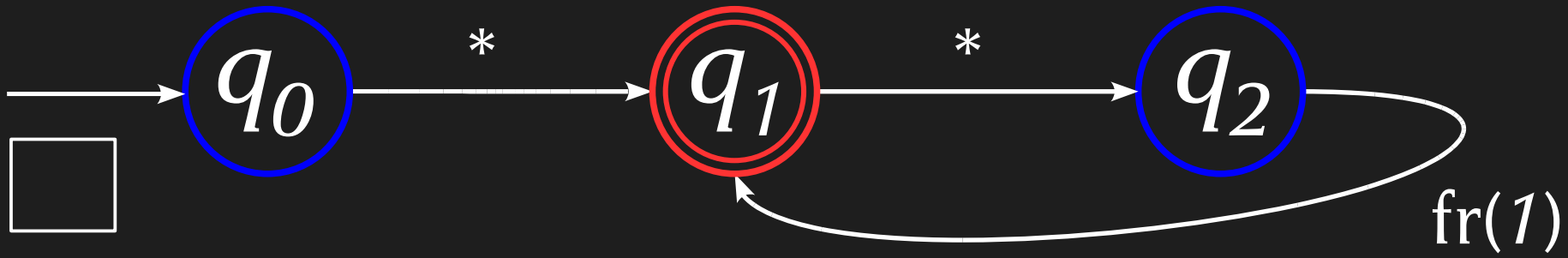
Fresh-register automata



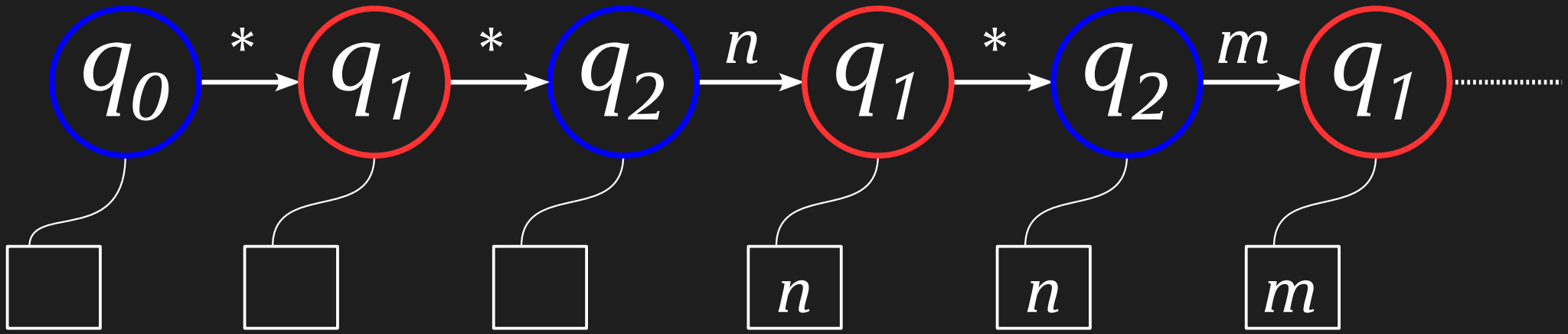


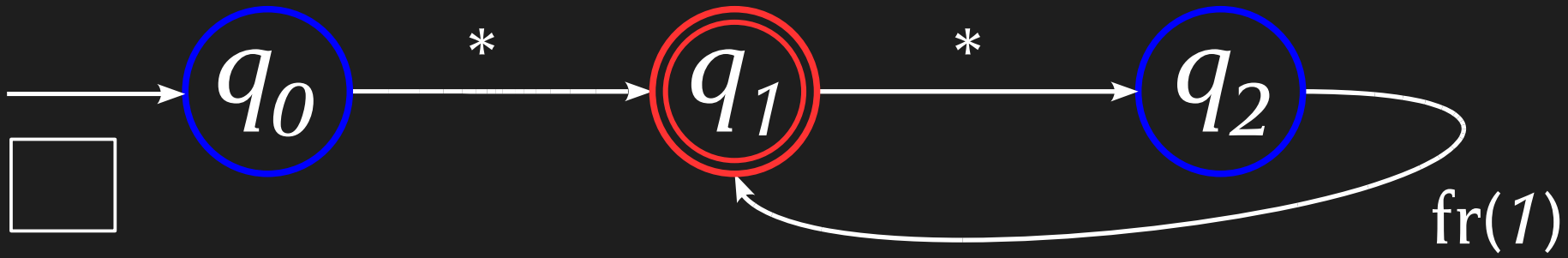
Fresh-register automata



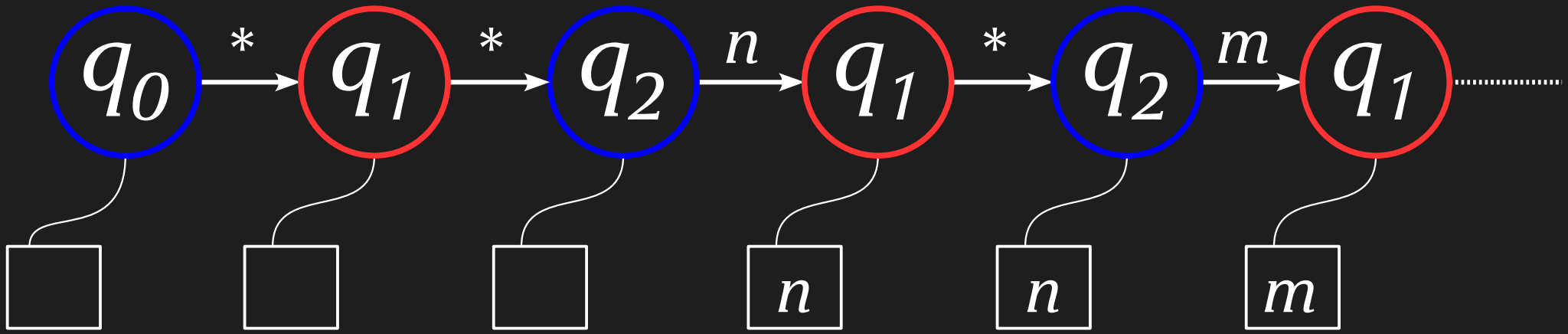


Fresh-register automata

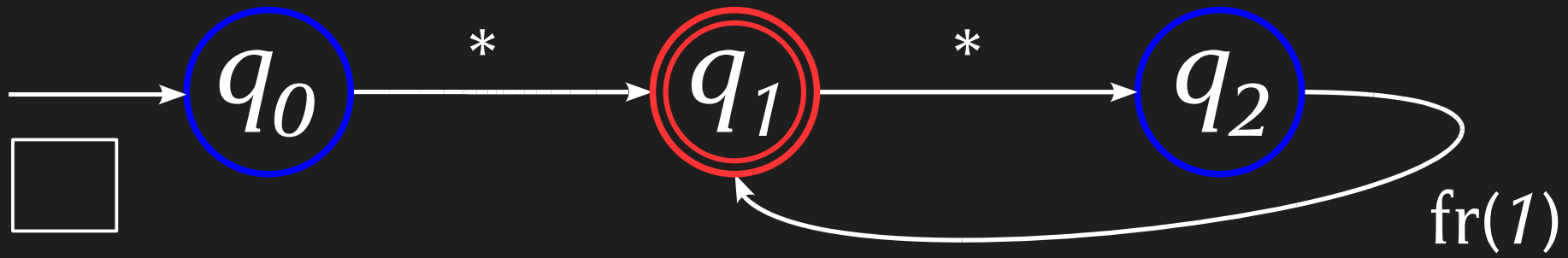




Fresh-register automata

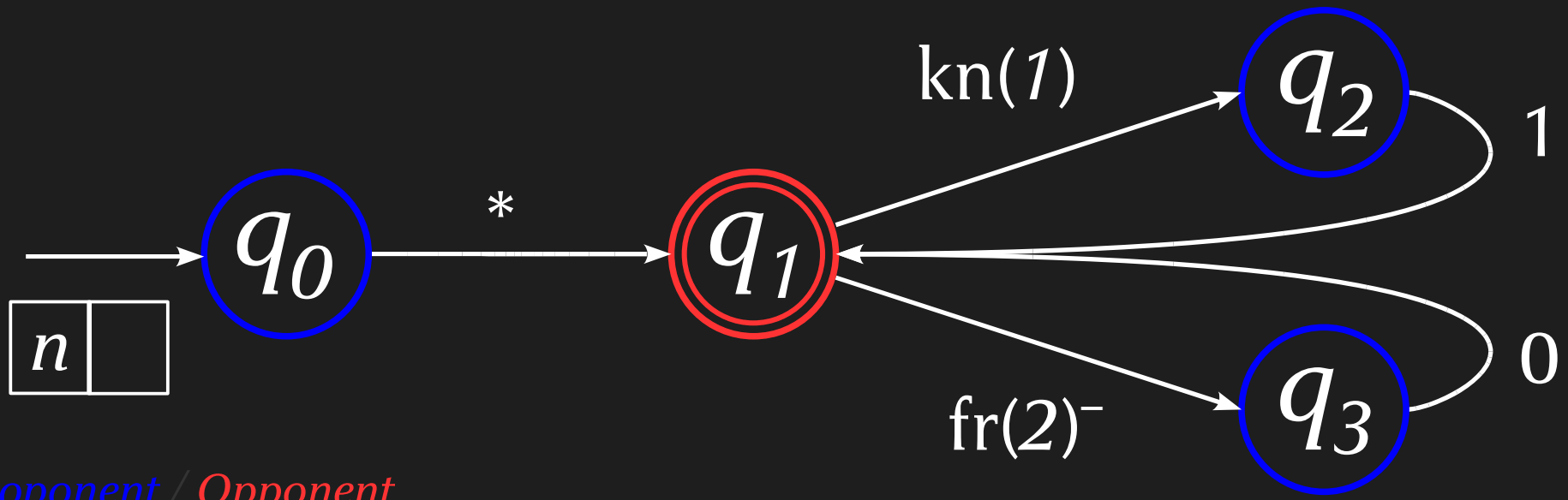


$\vdash \lambda z.\mathtt{ref}(0) : \mathtt{unit} \to \mathtt{intref}$



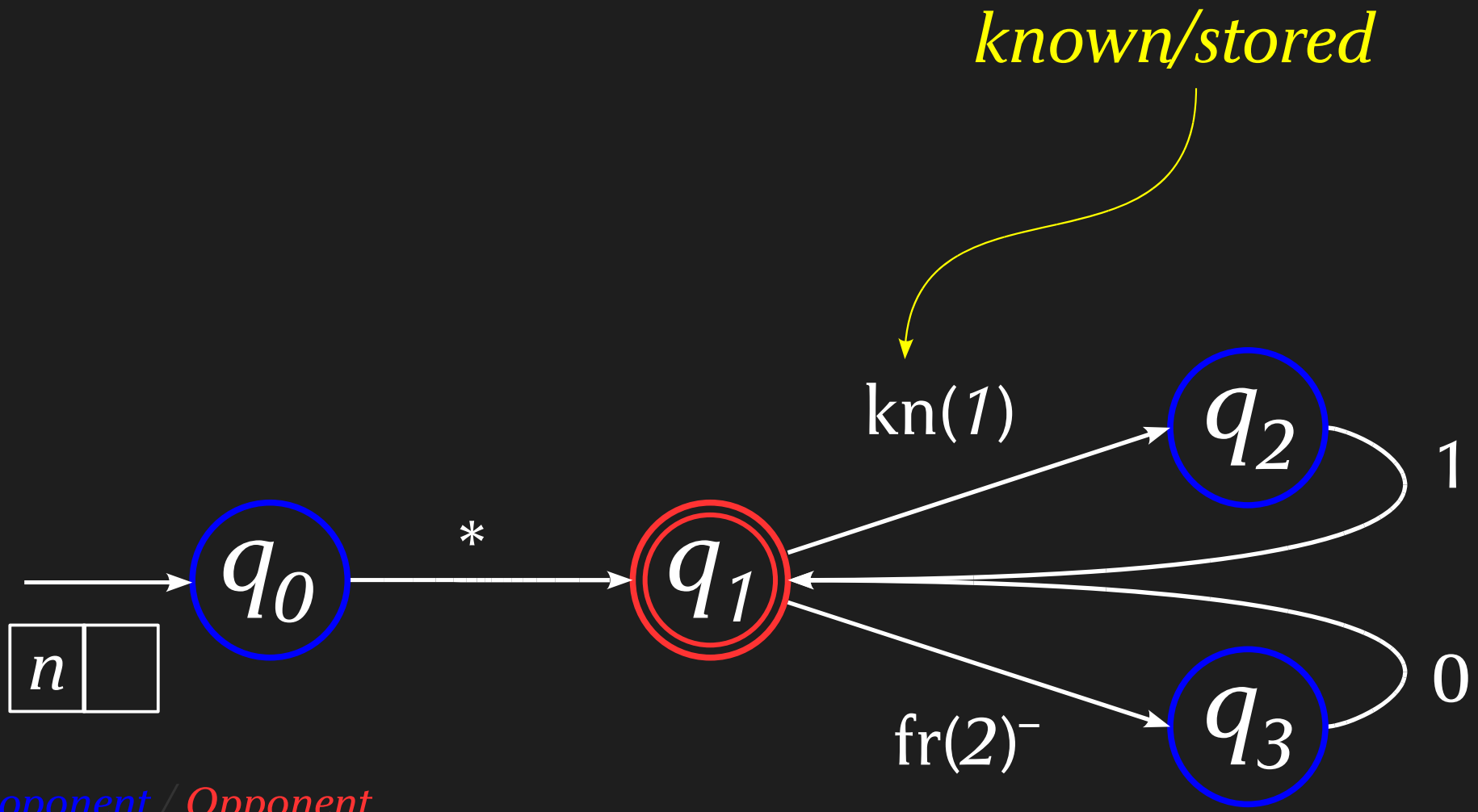
Proponent / *Opponent*

Fresh-register automata



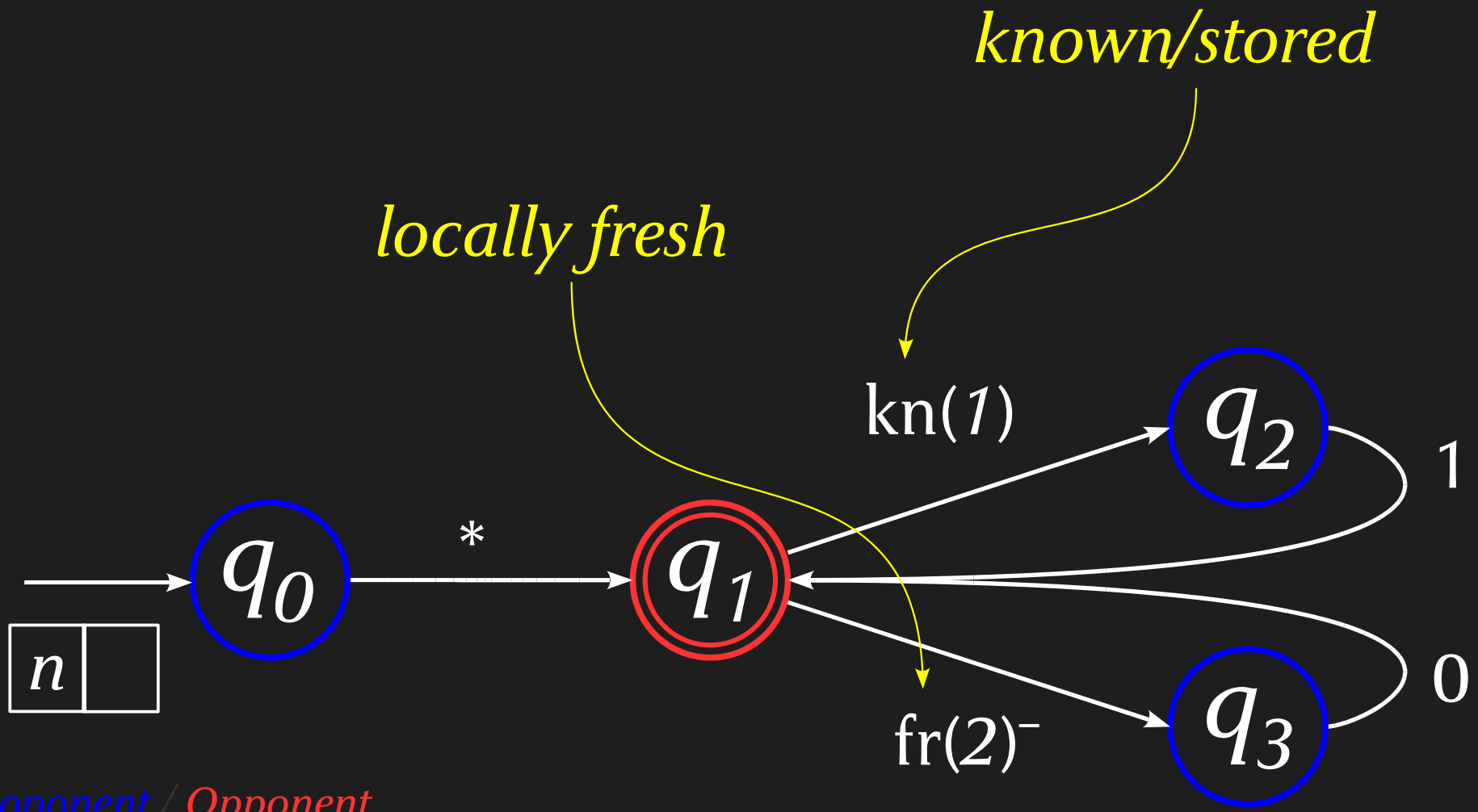
Proponent / *Opponent*

Fresh-register automata



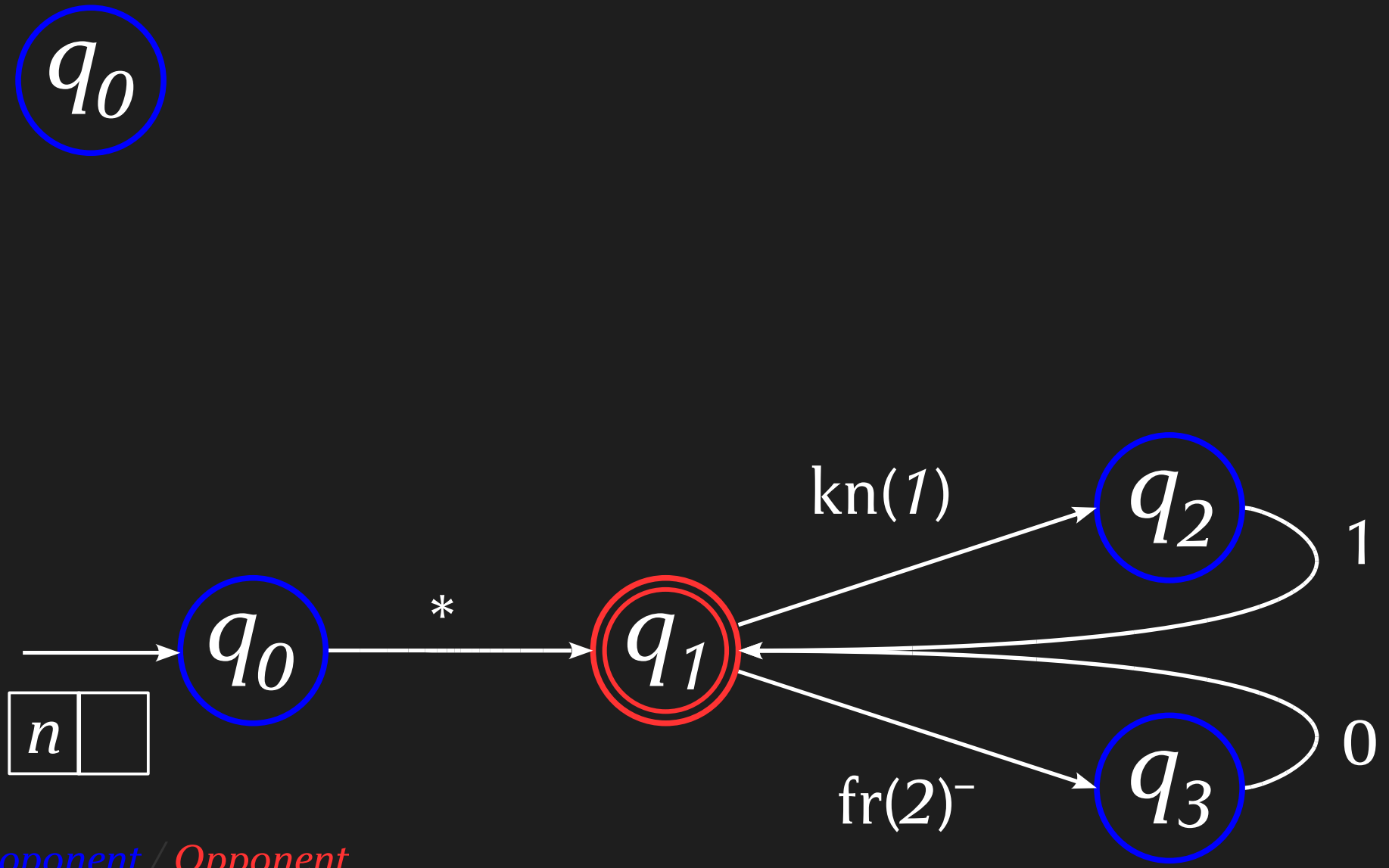
Proponent / *Opponent*

Fresh-register automata



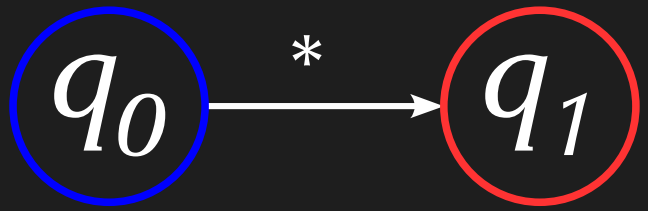
Proponent / *Opponent*

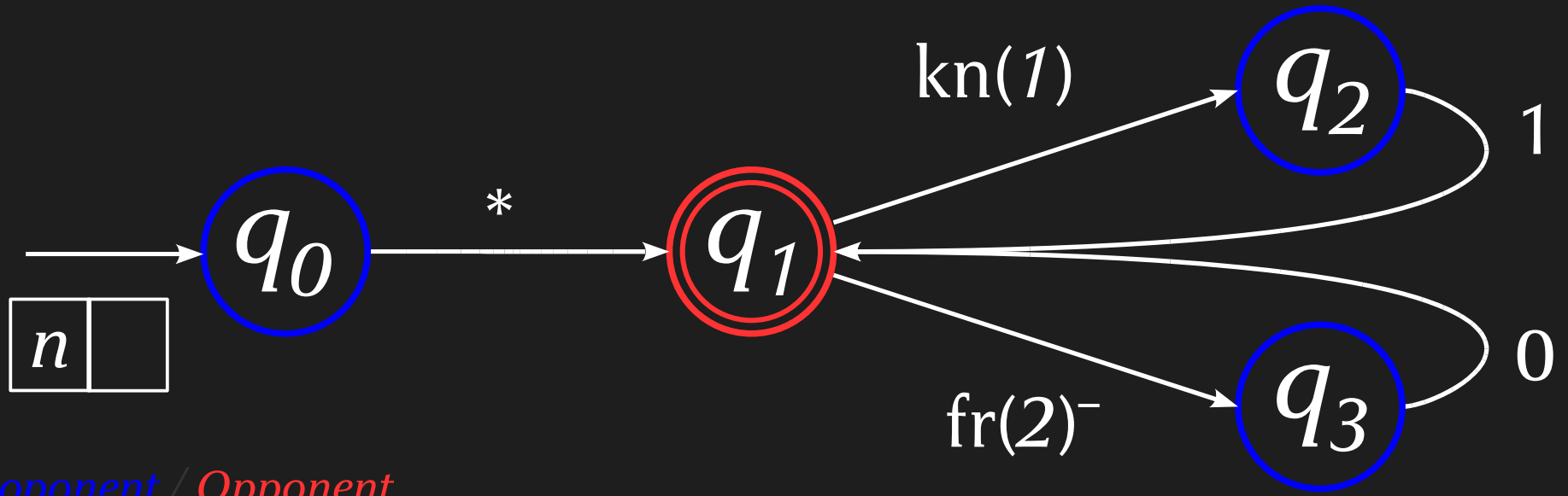
Fresh-register automata



Proponent / *Opponent*

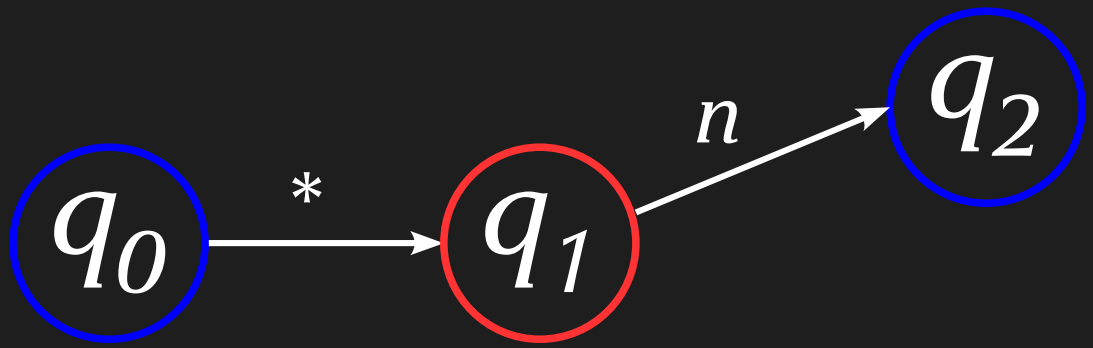
Fresh-register automata

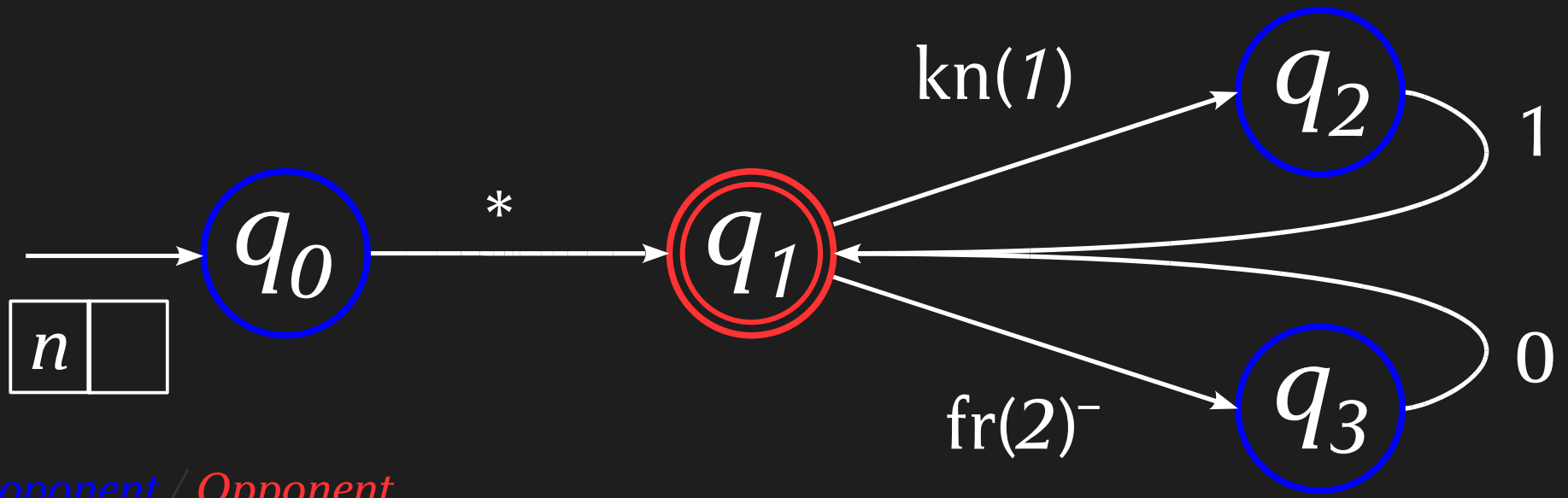




Proponent / *Opponent*

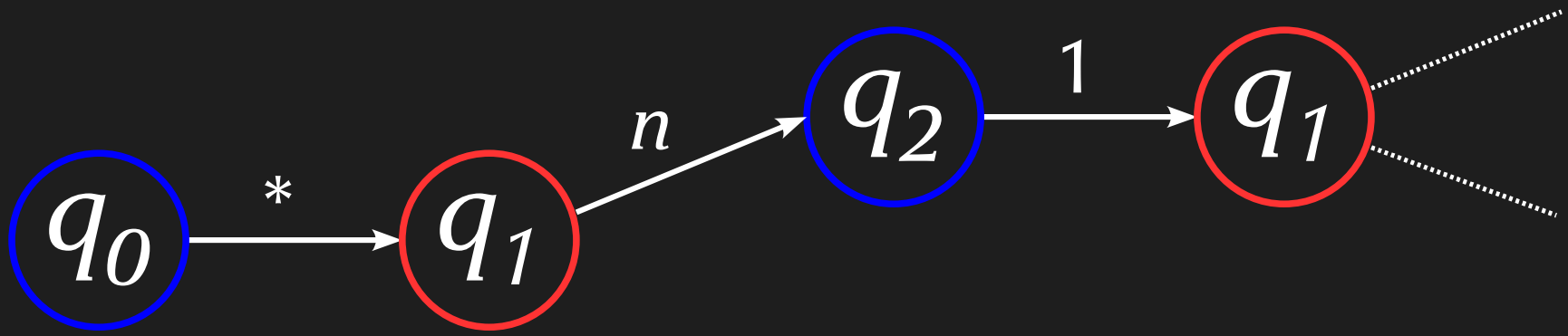
Fresh-register automata

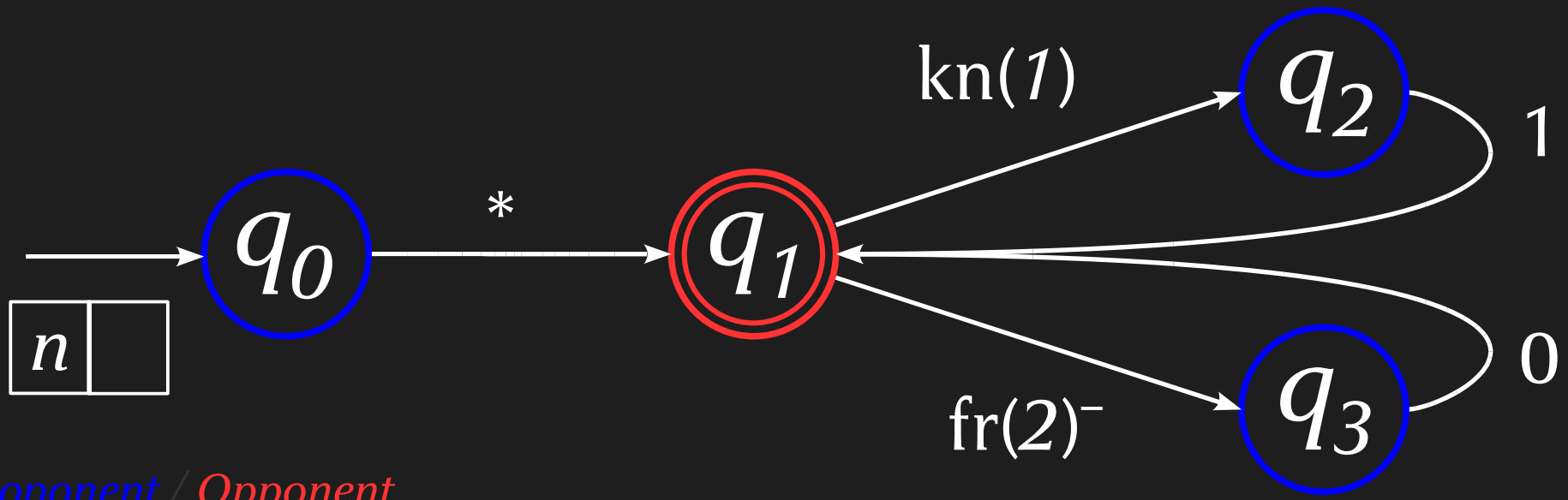




Proponent / *Opponent*

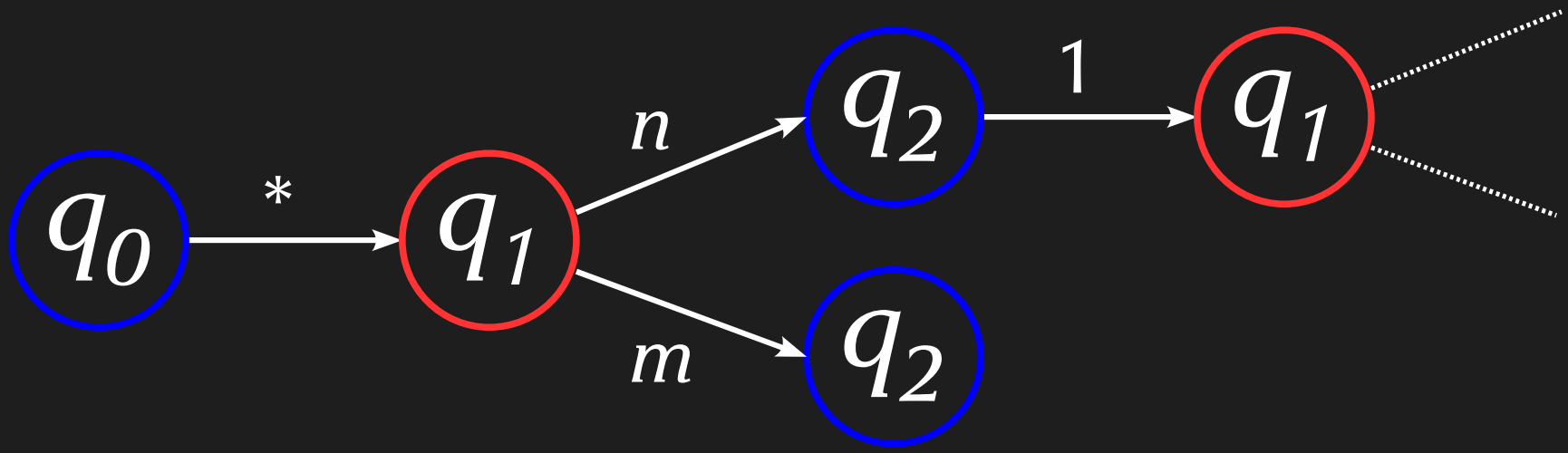
Fresh-register automata

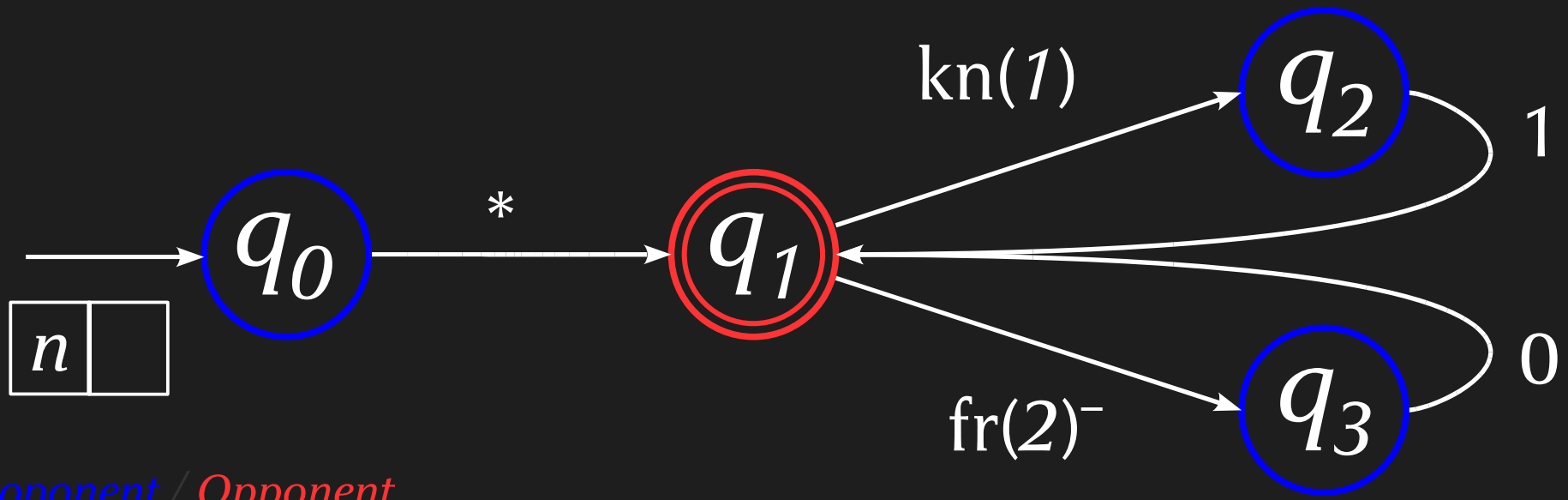




Proponent / *Opponent*

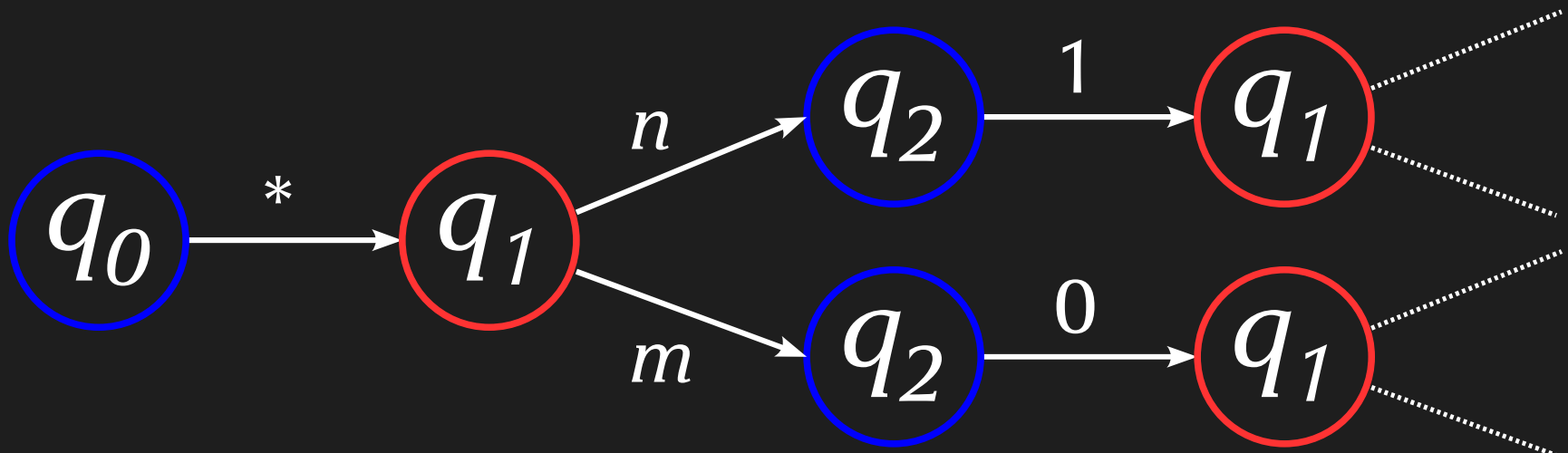
Fresh-register automata

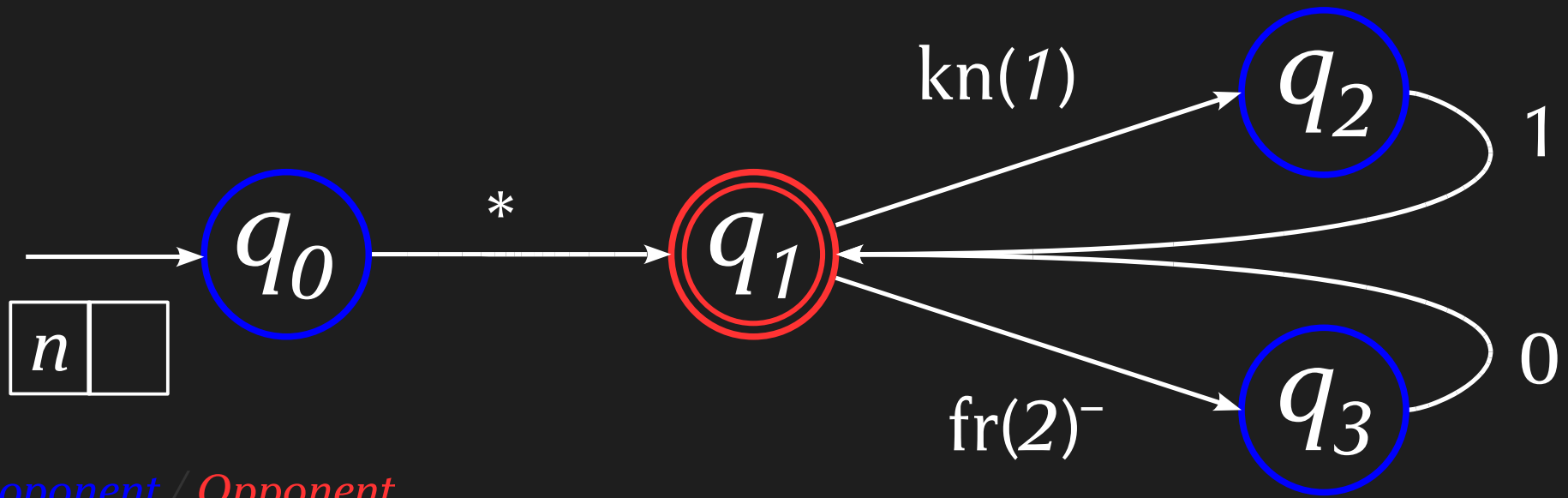




Proponent / *Opponent*

Fresh-register automata



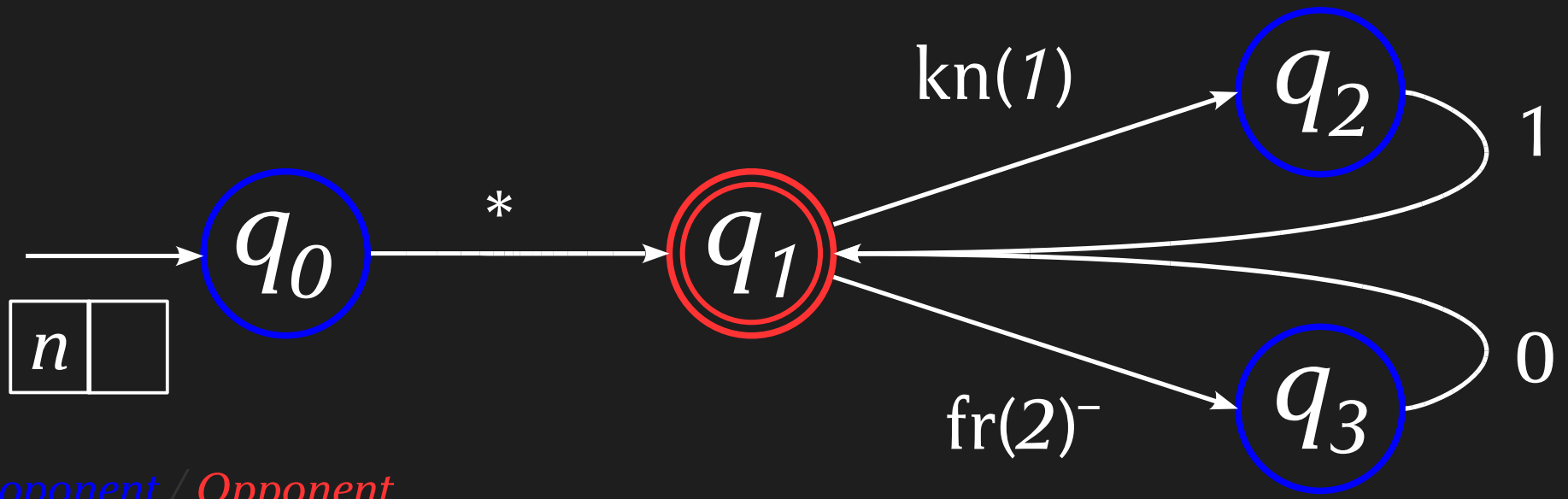


Proponent / *Opponent*

Fresh-register automata

$\vdash$ `let x = ref(0) in λy.(x == y) : intref → int`

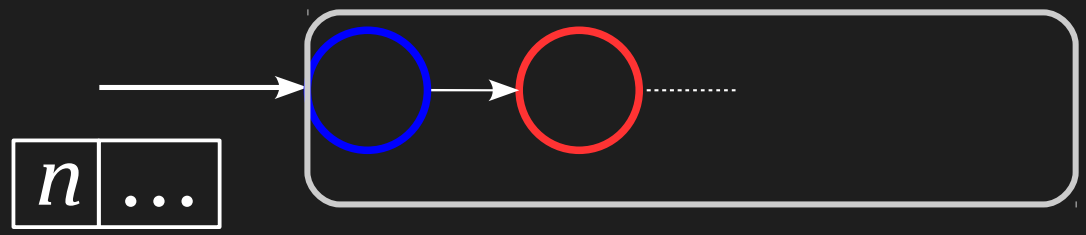
`x : intref` $\vdash$ `λy.(x == y) : intref → int`



Proponent / *Opponent*

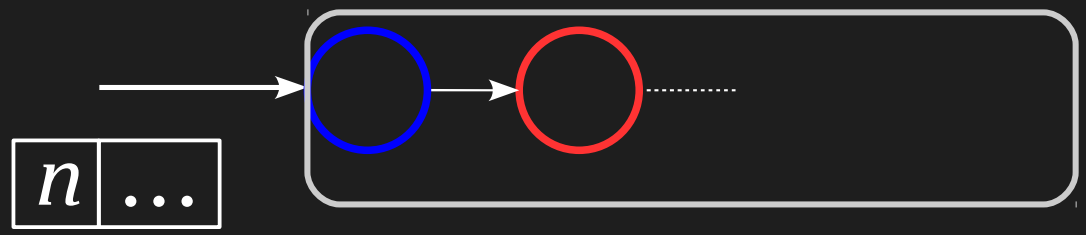
The *new* construction

$\texttt{x:intref},\Gamma \vdash M : \theta$



The *new* construction

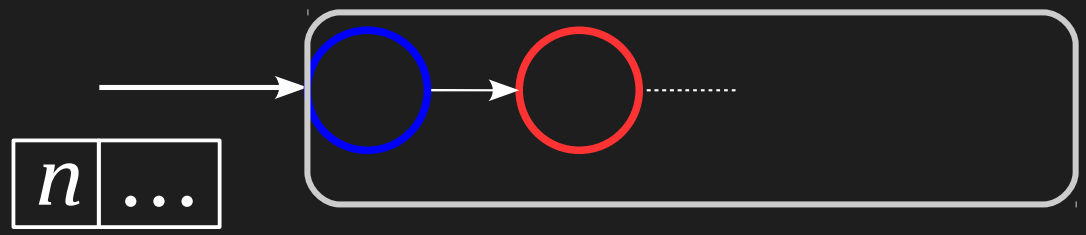
$\texttt{x:intref}, \Gamma \vdash M : \theta$



$\Gamma \vdash \texttt{let x = ref(0) in } M : \theta$

The *new* construction

$x : \mathtt{intref}, \Gamma \vdash M : \theta$

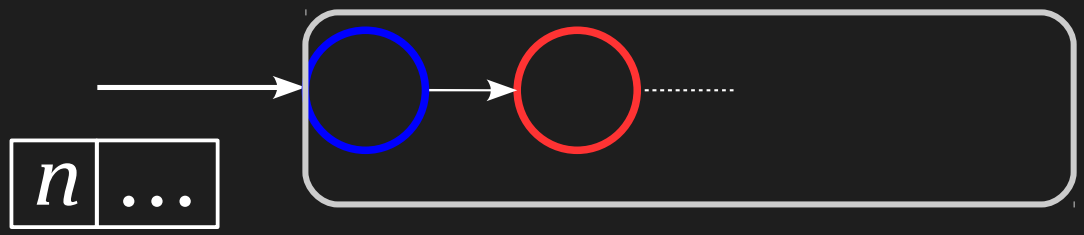


$\Gamma \vdash \mathtt{let}\ x = \mathtt{ref}(0)\ \mathtt{in}\ M : \theta$

Proponent / *Opponent*

The *new* construction

$\texttt{x:intref},\Gamma \vdash M : \theta$

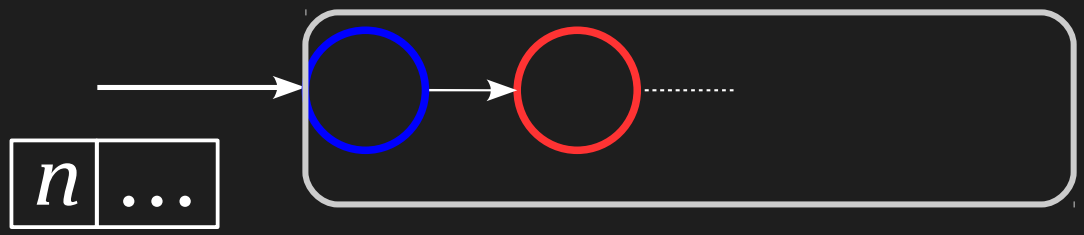


$\Gamma \vdash \texttt{let x = ref(0) in } M : \theta$

Proponent / *Opponent*

The *new* construction

$x{:}\texttt{intref}, \Gamma \vdash M : \theta$

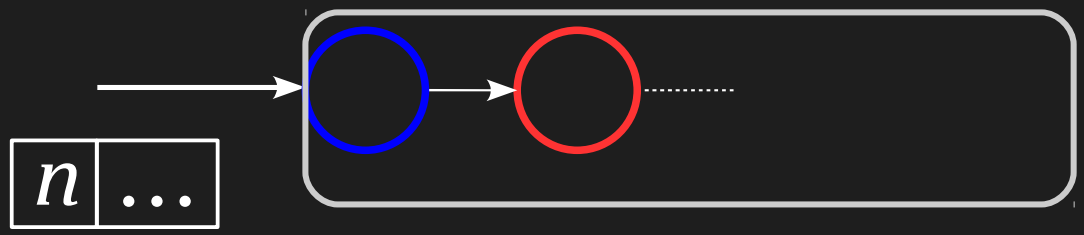


$\Gamma \vdash \texttt{let x = ref(0) in } M : \theta$

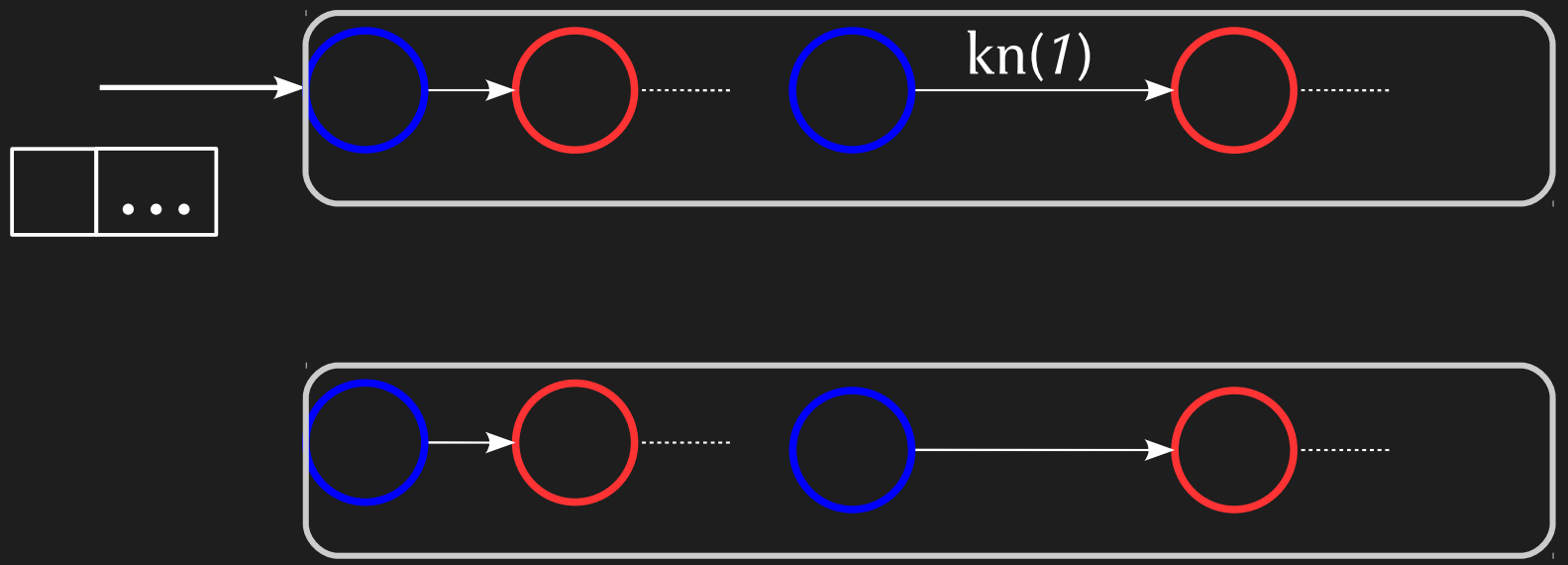
Proponent / *Opponent*

The *new* construction

$\texttt{x:intref},\Gamma \vdash M : \theta$



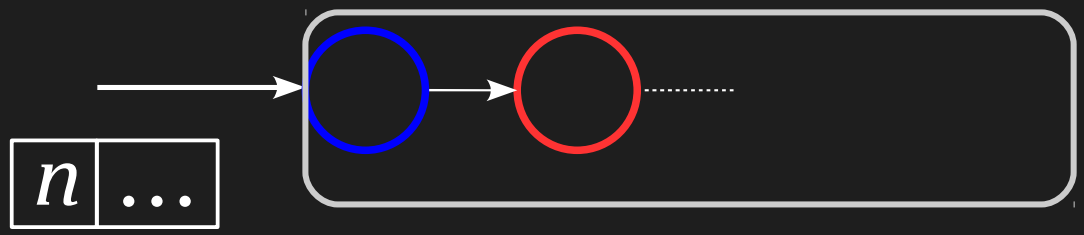
$\Gamma \vdash \texttt{let x = ref(0) in } M : \theta$



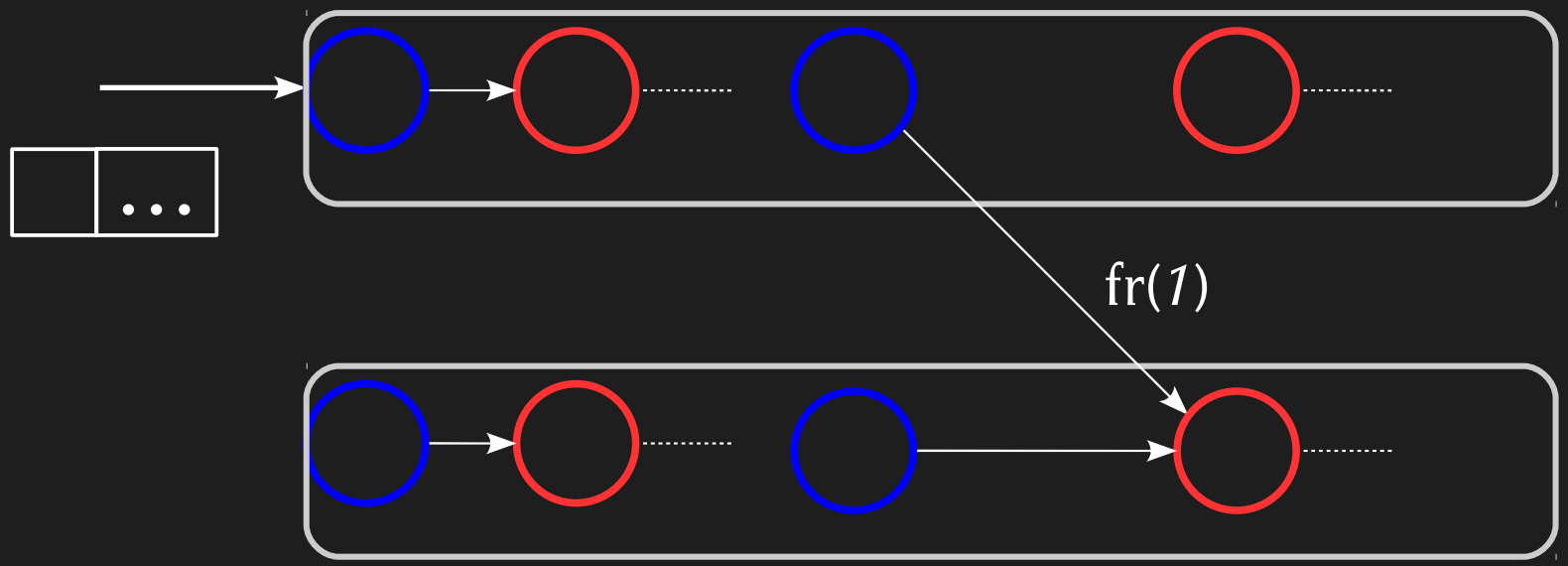
Proponent / *Opponent*

The *new* construction

$\texttt{x:intref},\Gamma \vdash M : \theta$



$\Gamma \vdash \texttt{let x = ref(0) in } M : \theta$

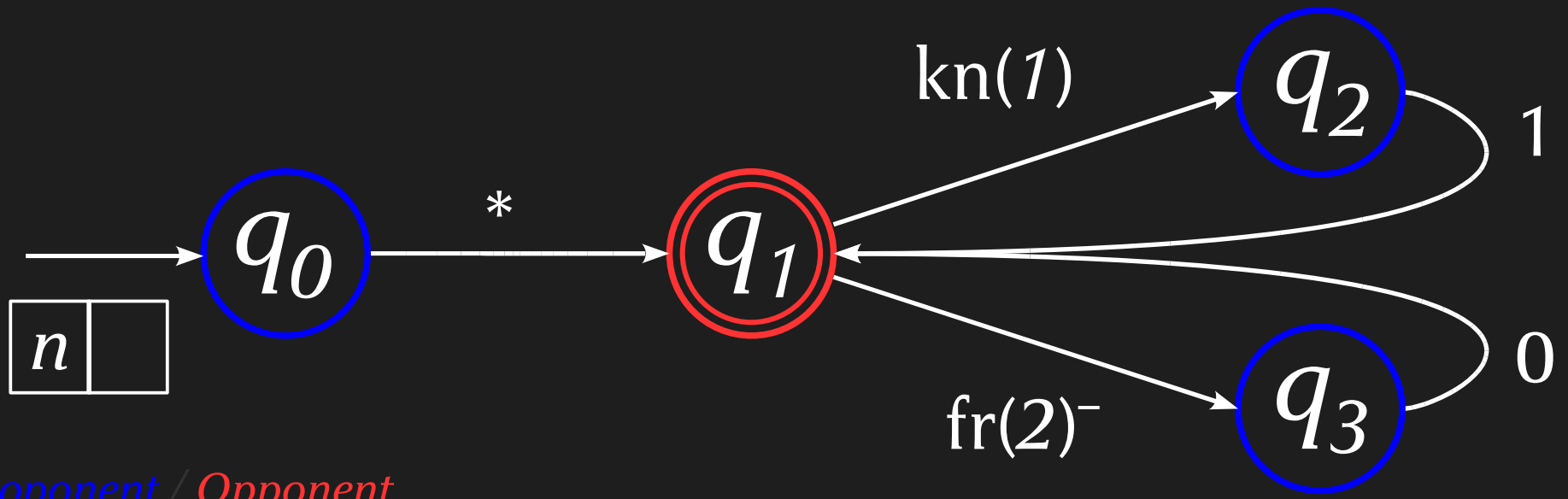


Proponent / *Opponent*

The *new* construction

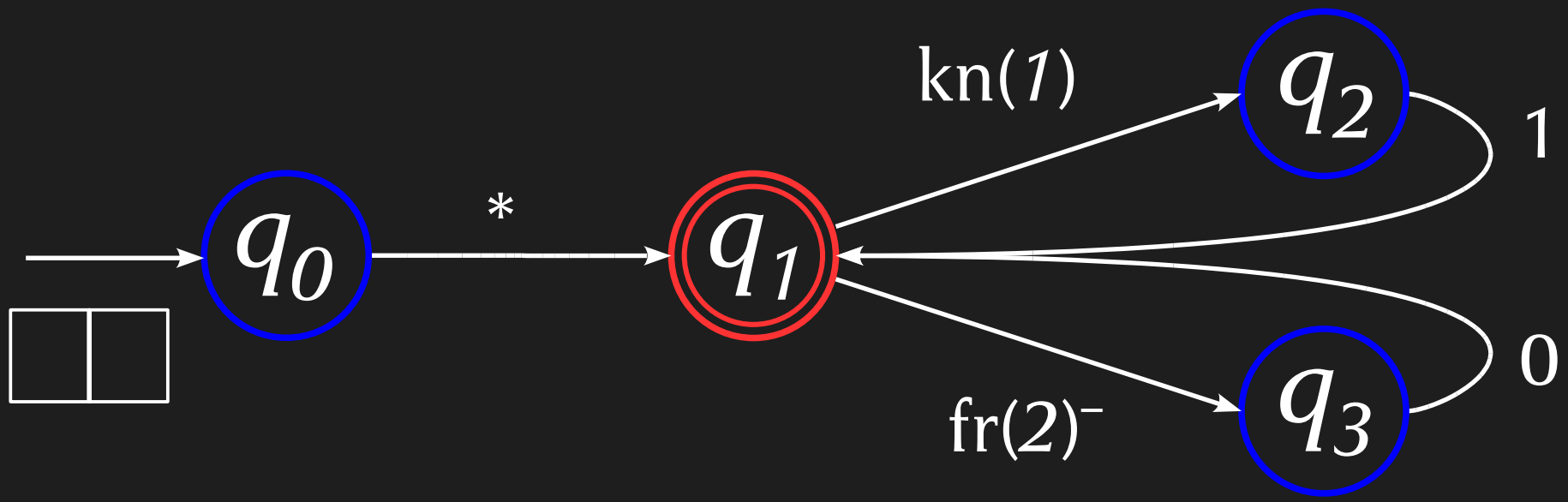
$\vdash$ `let x = ref(0) in λy.(x == y) : intref → int`

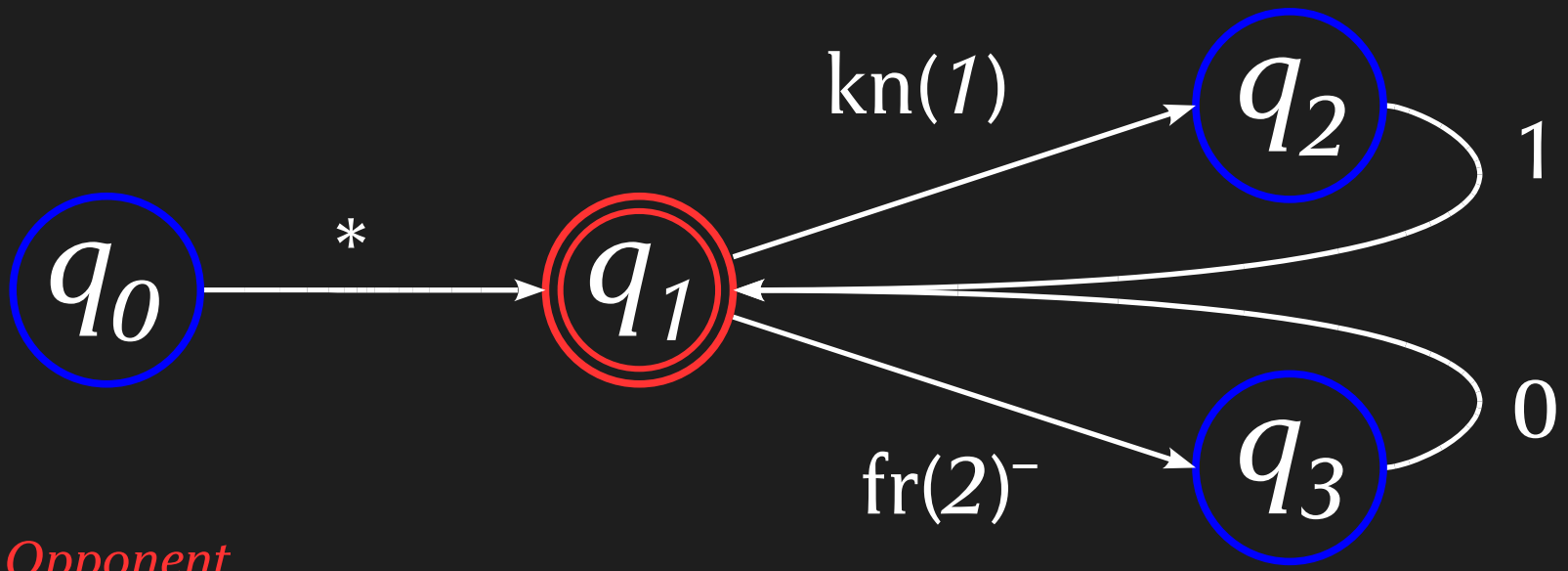
`x : intref` $\vdash$ `λy.(x == y) : intref → int`



Proponent / *Opponent*

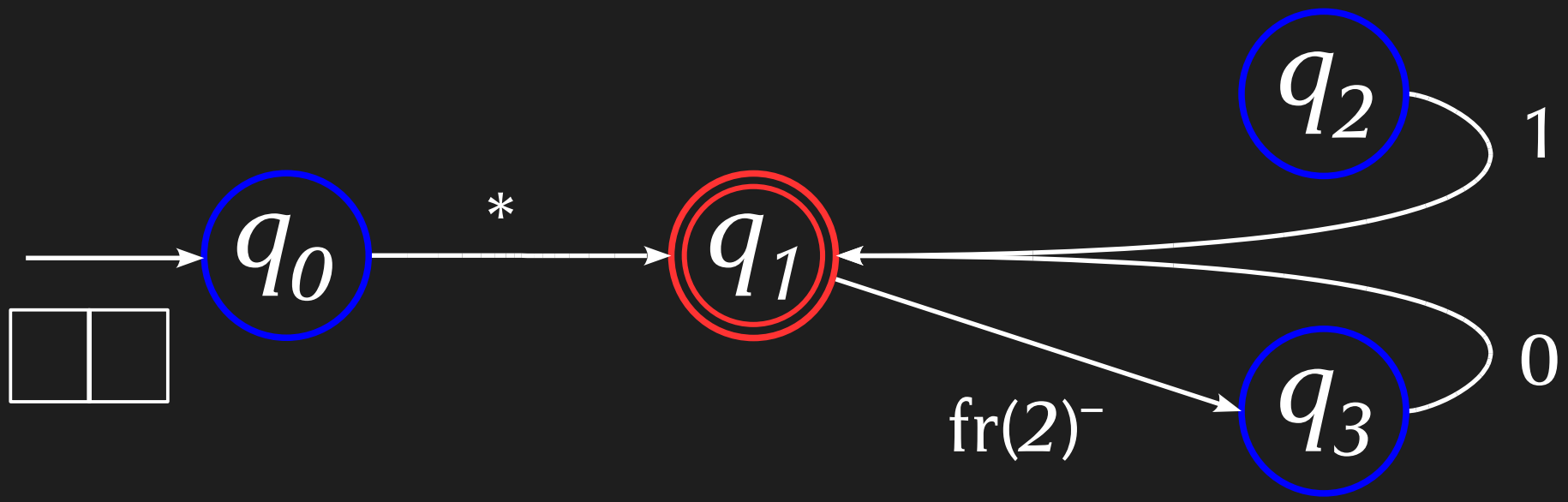
The *new* construction

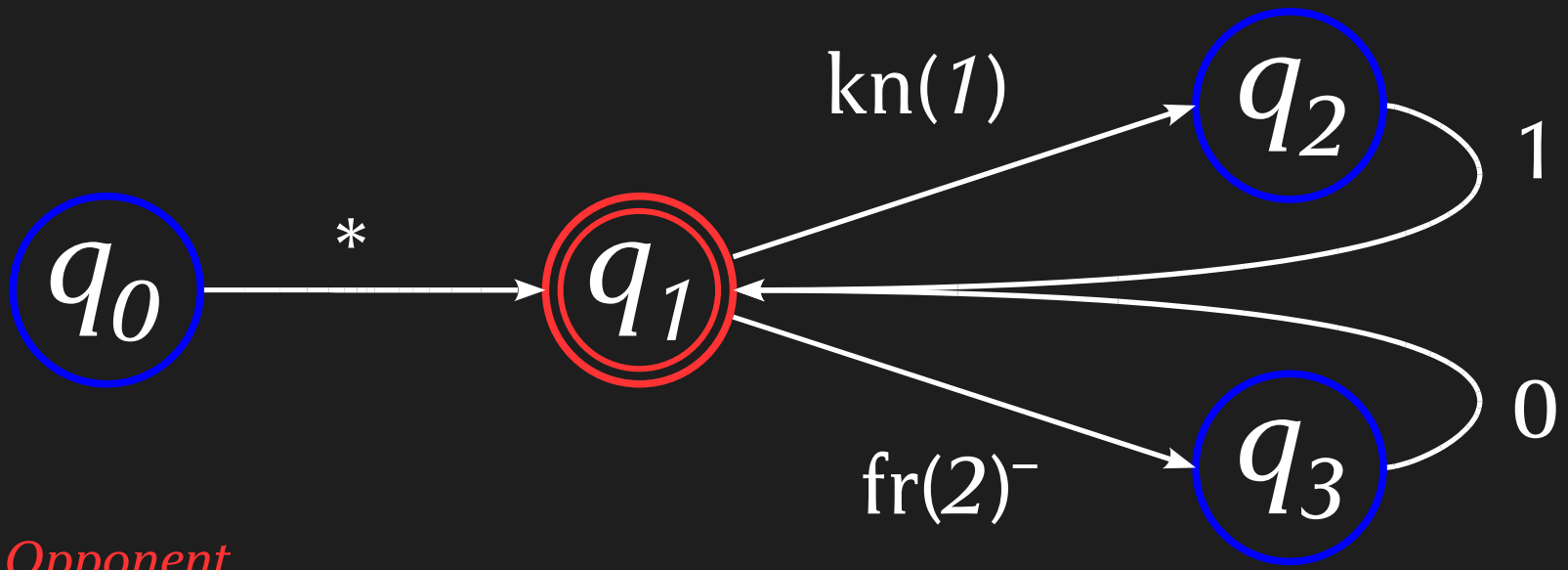




Proponent / *Opponent*

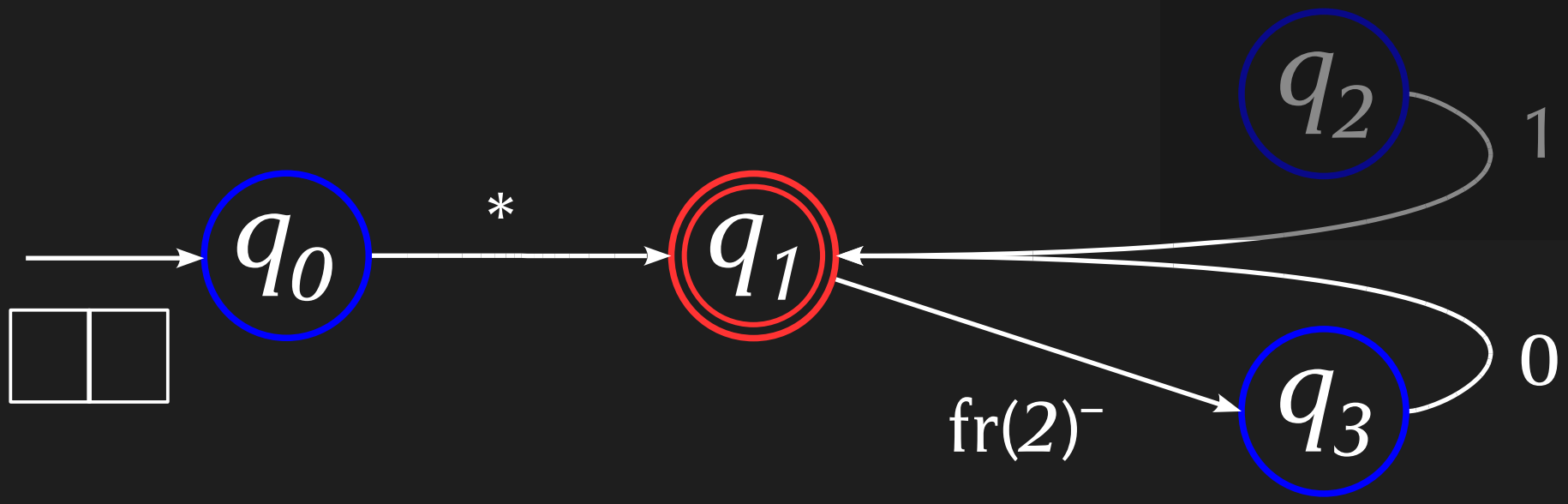
The *new* construction

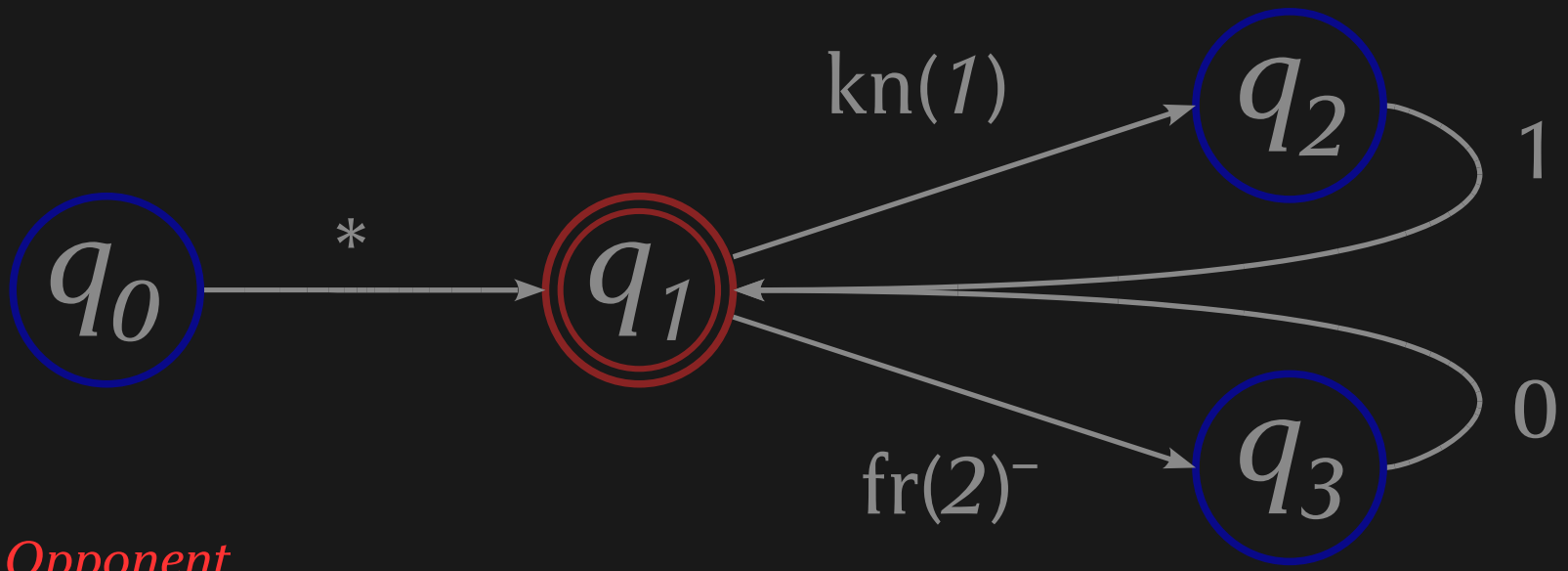




Proponent / *Opponent*

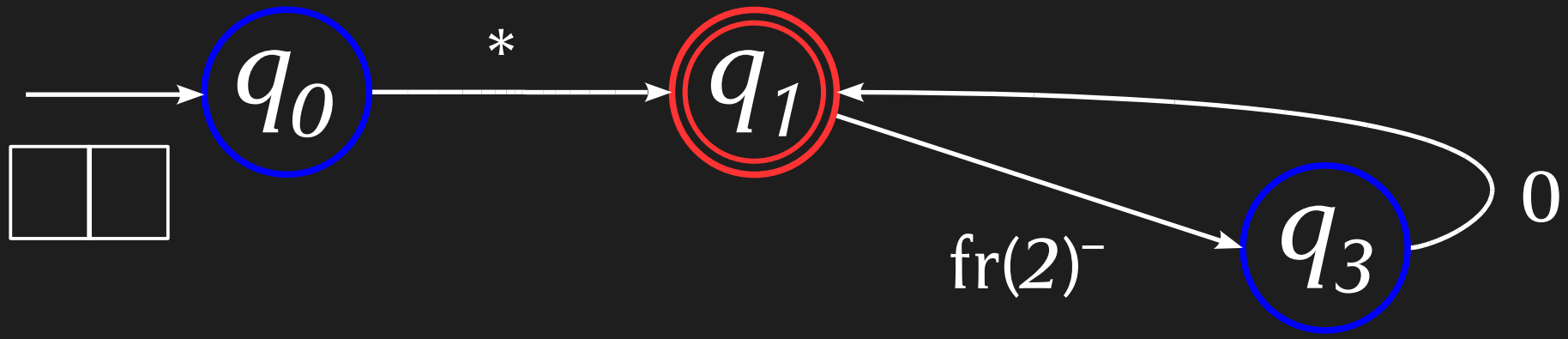
The *new* construction





Proponent / *Opponent*

The *new* construction



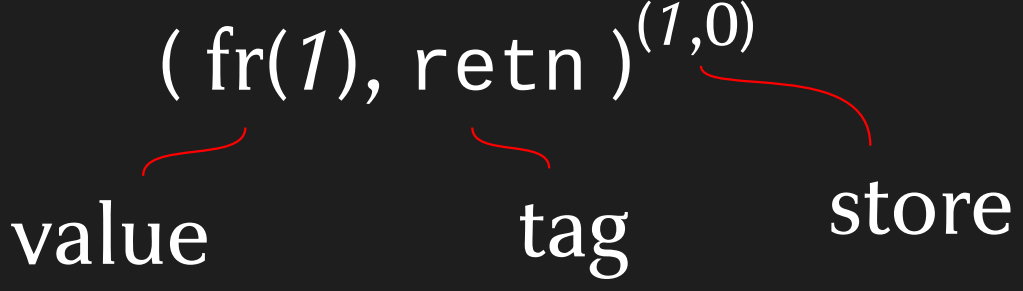
$\vdash$ `let x = ref(0) in λy.(x == y) : intref → int`

$\vdash$ `λy.0 : intref → int`

Proponent / *Opponent*

Automata for mini-RedML

- Labels are triples:



Automata for mini-RedML

- Labels are triples: $(\ \mathrm{fr}(\mathit{1}),\ \texttt{retn}\ )^{(\mathit{1},0)}$
- 1st construction: $[\![\mathrm{M}]\!] = \mathcal{L}(A_{\mathrm{M}})$

Automata for mini-RedML

- Labels are triples: $(\ \mathrm{fr}(1),\ \texttt{retn}\ )^{(1,0)}$
- 1st construction: $[\![\mathrm{M}]\!] = \mathcal{L}(A_{\mathrm{M}})$
- 2nd construction: $\overline{[\![\mathrm{M}]\!]} = \overline{[\![\mathrm{N}]\!]} \iff \overline{A_{\mathrm{M}}} \sim \overline{A_{\mathrm{N}}}$

Automata for mini-RedML

- Labels are triples: $(\ \mathrm{fr}(1),\ \texttt{retn}\ )^{(1,0)}$
- 1st construction: $[\![\mathrm{M}]\!] = \mathcal{L}(A_{\mathrm{M}})$
- 2nd construction: $\overline{\overline{[\![\mathrm{M}]\!]}} = \overline{\overline{[\![\mathrm{N}]\!]}} \iff \overline{\overline{A_{\mathrm{M}}}} \sim \overline{\overline{A_{\mathrm{N}}}}$
- Corollary: Program equivalence is decidable

$$\mathrm{M} \cong \mathrm{N} \iff \overline{\overline{[\![\mathrm{M}]\!]}} = \overline{\overline{[\![\mathrm{N}]\!]}} \iff \overline{\overline{A_{\mathrm{M}}}} \sim \overline{\overline{A_{\mathrm{N}}}}$$

Concluding

- Automata for generative games – programs
- What to do next:
  - Higher types
  - Complexity?
  - More expressiveness
  - Program verification

thank you!