Game Semantics

Andrzej S. Murawski and Nikos Tzevelekos

Lecture 1b: Introduction to Game Semantics

Sequentiality

The problem with **parallel-or** is that programs are not really functions.

For example, a program with two inputs:

- either does not use one of the inputs,
- or, if it does, it picks one to use first.

► Put otherwise, programs are **sequential computations**.

This is what makes behaviours like **parallel-or** non-programmable.

This mismatch between

- functions in domain-based models
- and programs in PCF

makes us look beyond functions for a model of programs.

Dynamic behaviours

Example. Consider a program

$$\textbf{count} : \text{int} \to \text{int}$$

such that:

- the first time we call it, it returns 1;
- the second time we call it, it returns 2;
- $\ldots$
- the i-th time we call it, it returns i.

While we cannot write such a program in PCF, it is easy to do it in any language like Java, Python, OCaML, etc.

▶ Programs can change their behaviour **dynamically**.

This is another source of mismatch between programs and functions.

Programs $\mapsto$ sequences of computation steps

Being sequential and dynamic, programs are best described as sequences of computation steps:

Programs $\mapsto$ sequences of computation steps

Being sequential and dynamic, programs are best described as sequences of computation steps:

left-or:

- call **left-or**(x, y)

Programs $\mapsto$ sequences of computation steps

Being sequential and dynamic, programs are best described as sequences of computation steps:

left-or:

- ○ call **left-or**(x, y)
- evaluate x

Programs $\mapsto$ sequences of computation steps

Being sequential and dynamic, programs are best described as sequences of computation steps:

left-or:

- call **left-or**(x, y)
- evaluate x
  - x is 1
  - return 1

Programs $\mapsto$ sequences of computation steps

Being sequential and dynamic, programs are best described as sequences of computation steps:

left-or:

- ○ call **left-or**(x, y)
- • evaluate x
  - ○ x is 1
  - • return 1

  - ○ x is 0
  - • evaluate y
    - ○ y is v
    - • return v

Programs $\mapsto$ (computation) games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

Programs $\mapsto$ (computation) games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

left-or:

- what is the result of **left-or**(x, y)?

Programs $\mapsto$ (computation) games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

left-or:

- ○ what is the result of **left-or**(x, y)?
- • what is the value of x?

Programs $\mapsto$ (computation) games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

left-or:

- ○ what is the result of **left-or**(x, y)?
- • what is the value of x?
  - ○ x is 1
  - • the result is 1

Programs $\mapsto$ (computation) games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

left-or:

- ○ what is the result of **left-or**(x, y)?
- ● what is the value of x?
  - ○ x is 1
  - ● the result is 1

  - ○ x is 0
  - ● what is the result of y?
    - ○ y is v
    - ● the result is v

Programs $\mapsto$ games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

Programs $\mapsto$ games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

$\vdash$ **count** : int $\rightarrow$ int :

- what is the result of **count**?

Programs $\mapsto$ games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

$\vdash$ **count** : int $\to$ int :

- ○ what is the result of **count**?
- ● it is a function

Programs $\mapsto$ games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

$\vdash$ **count** : int $\rightarrow$ int :

- ○ what is the result of **count**?
- • it is a function
  - ○ what is the result of the function on 42?

Programs $\mapsto$ games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

$\vdash$ **count** : int $\to$ int :

- ○ what is the result of **count**?
- • it is a function
  - ○ what is the result of the function on 42?
  - • it is 1

Programs $\mapsto$ games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

$\vdash$ **count** : int $\rightarrow$ int :

- ○ what is the result of **count**?
- ● it is a function
  - ○ what is the result of the function on 42?
  - ● it is 1
  - ○ what is the result of the function on 23?
  - ● it is 2

Programs $\mapsto$ games

Being sequential and dynamic, programs are best described as games (or dialogues) between the program and its calling context:

$\vdash$ **count** : int $\to$ int :

- ○ what is the result of **count**?
- ● it is a function
  - ○ what is the result of the function on 42?
  - ● it is 1
  - ○ what is the result of the function on 23?
  - ● it is 2
  - ○ what is the result of the function on 25?
  - ● it is 3

  ...

Higher-order programs

Example. Consider a program

$$f : \mathsf{int} \to \mathsf{int} \;\vdash\; \textbf{plusOne} : \mathsf{int} \to \mathsf{int}$$

given by: $\textbf{plusOne} \;\equiv\; \lambda x^{\mathsf{int}}.\, f x + 1$

- if we call it with a function f
- it returns a function that, on input x, returns $f(x) + 1$

Higher-order programs

Example. Consider a program

$$f : \mathsf{int} \to \mathsf{int} \;\vdash\; \textbf{plusOne} : \mathsf{int} \to \mathsf{int}$$

given by: $\textbf{plusOne} \;\equiv\; \lambda x^{\mathsf{int}}.\, f x + 1$

- if we call it with a function f
- it returns a function that, on input x, returns $f(x) + 1$

E.g. taking $\mathbf{2x} \equiv \lambda y^{\mathsf{int}}.\, 2 * y : \mathsf{int} \to \mathsf{int}$:

$$\begin{aligned}(\lambda f.\textbf{plusOne})\,\mathbf{2x}\,0 \;&\longrightarrow\; (\lambda x^{\mathsf{int}}.\mathbf{2x}\,x + 1)\,0\\ &\longrightarrow\; \mathbf{2x}\,0 + 1 \;\longrightarrow^*\; 1\end{aligned}$$

$$\begin{aligned}(\lambda f.\textbf{plusOne})\,\mathbf{2x}\,42 \;&\longrightarrow\; (\lambda x^{\mathsf{int}}.\mathbf{2x}\,x + 1)\,42\\ &\longrightarrow\; \mathbf{2x}\,42 + 1 \;\longrightarrow^*\; 85\end{aligned}$$

Higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$

given by: **plusOne** $\equiv \lambda x^{\mathsf{int}}. f x + 1.$

Game:

- given function f, what is the result of **plusOne**?
- it is a function f'

Higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$

given by: **plusOne** $\equiv \lambda x^{\mathsf{int}}. f x + 1.$

Game:

- given function f, what is the result of **plusOne**?
- it is a function f'
  - what is the result of f' on 42?

Higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$

given by: **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1.$

Game:

- ○ given function f, what is the result of **plusOne**?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?

Higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \ \vdash \ \textbf{plusOne} : \mathsf{int} \to \mathsf{int}$

given by: $\textbf{plusOne} \ \equiv \ \lambda x^{\mathsf{int}}.\, f x + 1.$

Game:

- ○ given function f, what is the result of **plusOne**?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 84

Higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$

given by: **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1$.

Game:

- ○ given function f, what is the result of **plusOne**?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 84
  - • it is 85

  . . .

Higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$

given by: **plusOne** $\equiv \lambda x^{\mathsf{int}}. f x + 1.$

Game:

- ○ given function f, what is the result of **plusOne**?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?

Higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$

given by: **plusOne** $\equiv \lambda x^{\mathsf{int}}. f x + 1.$

Game:

- ○ given function f, what is the result of **plusOne**?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 35

Higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$

given by: **plusOne** $\equiv \lambda x^{\mathsf{int}}. f x + 1.$

Game:

- ○ given function f, what is the result of **plusOne**?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 35
  - • it is 71

  . . .

More higher-order programs

Example. Consider a program

$$f : \mathsf{int} \to \mathsf{int} \;\vdash\; \mathbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$$

given by: $\mathbf{plusOne}^2 \;\equiv\; \lambda x^{\mathsf{int}}.\, f(fx+1)+1$

- if we call it with a function f
- it returns a function that, on input x, returns $f(f(x)+1)+1$

More higher-order programs

Example. Consider a program

$$f : \mathsf{int} \to \mathsf{int} \ \vdash \ \mathbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$$

given by: $\mathbf{plusOne}^2 \ \equiv \ \lambda x^{\mathsf{int}}.\, f(fx+1)+1$

- if we call it with a function f
- it returns a function that, on input x, returns $f(f(x)+1)+1$

E.g. taking $\mathbf{2x} \equiv \lambda y^{\mathsf{int}}.\, 2 * y : \mathsf{int} \to \mathsf{int}$:

$$(\lambda f.\mathbf{plusOne}^2)\,\mathbf{2x}\,0 \quad \longrightarrow \quad (\lambda x^{\mathsf{int}}.\,\mathbf{2x}(\mathbf{2x}\,x+1)+1)\,0$$
$$\longrightarrow \quad \mathbf{2x}(\mathbf{2x}\,0+1)+1 \qquad \longrightarrow^* \quad 1$$

$$(\lambda f.\mathbf{plusOne}^2)\,\mathbf{2x}\,42 \quad \longrightarrow \quad (\lambda x^{\mathsf{int}}.\,\mathbf{2x}(\mathbf{2x}\,x+1)+1)\,42$$
$$\longrightarrow \quad \mathbf{2x}(\mathbf{2x}\,42+1)+1 \qquad \longrightarrow^* \quad 171$$

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash \mathbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$ given by: $\mathbf{plusOne}^2 \equiv \lambda x^{\mathsf{int}}. f(f x + 1) + 1$.

Game:

- ○ given function f, what is the result of $\mathbf{plusOne}^2$?
- ● it is a function f'

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ $\mathbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$ given by: $\mathbf{plusOne}^2 \equiv \lambda x^{\mathsf{int}}. f(fx+1)+1.$

Game:

- given function f, what is the result of $\mathbf{plusOne}^2$?
- it is a function f'
  - what is the result of f' on 42?

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash \mathbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$ given by: $\mathbf{plusOne}^2 \equiv \lambda x^{\mathsf{int}}. f(f x + 1) + 1$.

Game:

- ○ given function f, what is the result of $\mathbf{plusOne}^2$?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash \textbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$ given by: $\textbf{plusOne}^2 \equiv \lambda x^{\mathsf{int}}. f(fx + 1) + 1$.

Game:

- ○ given function f, what is the result of $\textbf{plusOne}^2$?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 84

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash \textbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$

given by: $\textbf{plusOne}^2 \equiv \lambda x^{\mathsf{int}}.\, f(f x + 1) + 1$.

Game:

- ○ given function f, what is the result of $\textbf{plusOne}^2$?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 84
    - • what is the result of f on 85?

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash \mathbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$ given by: $\mathbf{plusOne}^2 \equiv \lambda x^{\mathsf{int}}. f(fx+1)+1.$

Game:

- ○ given function f, what is the result of **plusOne**^{2}?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 84
    - • what is the result of f on 85?
    - ○ it is 170

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne**$^2 : \mathsf{int} \to \mathsf{int}$ given by: **plusOne**$^2 \equiv \lambda x^{\mathsf{int}}. f(f x + 1) + 1$.

Game:

- ○ given function f, what is the result of **plusOne**^{2}?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 84
    - • what is the result of f on 85?
    - ○ it is 170
  - • it is 171

  . . .

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne**$^2 : \mathsf{int} \to \mathsf{int}$

given by: **plusOne**$^2 \equiv \lambda x^{\mathsf{int}}. f(fx + 1) + 1.$

Game:

- ○ given function f, what is the result of **plusOne**^{2}?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne**$^2 : \mathsf{int} \to \mathsf{int}$ given by: **plusOne**$^2 \equiv \lambda x^{\mathsf{int}}. f(fx+1)+1.$

Game:

- ○ given function f, what is the result of **plusOne**^{2}?
- ● it is a function f'
  - ○ what is the result of f' on 42?
    - ● what is the result of f on 42?
    - ○ it is 35

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash \textbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$ given by: $\textbf{plusOne}^2 \equiv \lambda x^{\mathsf{int}}. f(f x + 1) + 1$.

Game:

- ○ given function f, what is the result of $\textbf{plusOne}^2$?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 35
    - • what is the result of f on 36?

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ $\mathbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$ given by: $\mathbf{plusOne}^2 \equiv \lambda x^{\mathsf{int}}. f(f x + 1) + 1.$

Game:

- ○ given function f, what is the result of $\mathbf{plusOne}^2$?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 35
    - • what is the result of f on 36?
    - ○ it is 15

More higher-order games

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash \textbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}$
given by: $\textbf{plusOne}^2 \equiv \lambda x^{\mathsf{int}}.\, f(f x + 1) + 1.$

Game:

- ○ given function f, what is the result of **plusOne**^{2}?
- • it is a function f'
  - ○ what is the result of f' on 42?
    - • what is the result of f on 42?
    - ○ it is 35
    - • what is the result of f on 36?
    - ○ it is 15
  - • it is 16

  . . .

Anatomy of games

We describe programs as **games**. More precisely:

- games are sequences of **moves**, called **plays**
- following some **formal conditions**:

Anatomy of games

We describe programs as **games**. More precisely:

- games are sequences of **moves**, called **plays**
- following some **formal conditions**:
  - games are played between **two players**:
    - **Opponent (O)**, that represents the program's **context**
    - **Proponent (P)**, that represents the **program**

Revisit count

Being sequential and dynamic, programs are best described as games between a Proponent (P) and an Opponent (O):

Revisit count

Being sequential and dynamic, programs are best described as games between a Proponent (P) and an Opponent (O):

$\vdash$ **count** : int $\to$ int :

O what is the result of **count**?

P it is a function

- O what is the result of the function on 42?
- P it is 1
- O what is the result of the function on 23?
- P it is 2
- O what is the result of the function on 25?
- P it is 3
- ...

Anatomy of games

We describe programs as **games**. More precisely:

- games are sequences of **moves**, called **plays**
- following some **formal conditions**:
  - games are played between **two players**:
    - **Opponent (O)**, that represents the program's **context**
    - **Proponent (P)**, that represents the **program**

Anatomy of games

We describe programs as **games**. More precisely:

- games are sequences of **moves**, called **plays**
- following some **formal conditions**:
  - games are played between **two players**:
    - **Opponent (O)**, that represents the program's **context**
    - **Proponent (P)**, that represents the **program**
  - P and O alternate; O starts first

Anatomy of games

We describe programs as **games**. More precisely:

- games are sequences of **moves**, called **plays**
- following some **formal conditions**:
  - games are played between **two players**:
    - **Opponent (O)**, that represents the program's **context**
    - **Proponent (P)**, that represents the **program**
  - P and O alternate; O starts first
- moves come in two forms:
  - moves that call functions are **questions (Q)**
  - moves that return function calls are **answers (A)**

Revisit count

Being sequential and dynamic, programs are best described as games between a Proponent (P) and an Opponent (O):

Revisit count

Being sequential and dynamic, programs are best described as games between a Proponent (P) and an Opponent (O):

$\vdash$ **count** : int $\rightarrow$ int :

OQ what is the result of **count**?

PA it is a function

- OQ what is the result of the function on 42?
- PA it is 1
- OQ what is the result of the function on 23?
- PA it is 2
- OQ what is the result of the function on 25?
- PA it is 3
- ...

Anatomy of games

We describe programs as **games**. More precisely:

- games are sequences of **moves**, called **plays**
- following some **formal conditions**:
  - games are played between **two players**:
    - **Opponent (O)**, that represents the program's **context**
    - **Proponent (P)**, that represents the **program**
  - P and O alternate; O starts first
  - moves come in two forms:
    - moves that call functions are **questions (Q)**
    - moves that return function calls are **answers (A)**

Anatomy of games

We describe programs as **games**. More precisely:

- games are sequences of **moves**, called **plays**
- following some **formal conditions**:
  - games are played between **two players**:
    - **Opponent (O)**, that represents the program's **context**
    - **Proponent (P)**, that represents the **program**
  - P and O alternate; O starts first
  - moves come in two forms:
    - moves that call functions are **questions (Q)**
    - moves that return function calls are **answers (A)**
  - each A refers to a Q by the opposite player
  - ...

Games more formally

$[\![\vdash$ **count** $: \text{int} \to \text{int}]\!]$ is a set of plays of the form:

$$\begin{array}{cccccccccc} \star & \dagger & 42 & 1 & 23 & 2 & 25 & 3 & \cdots \\ \textit{OQ} & \textit{PA} & \textit{OQ} & \textit{PA} & \textit{OQ} & \textit{PA} & \textit{OQ} & \textit{PA} & \end{array}$$

where:

- the first move is played by O and asks the result of **count**, given an empty context (so, $\star$ is a move representing the empty context)

Games more formally

$[\![\vdash \textbf{count} : \text{int} \to \text{int}]\!]$ is a set of plays of the form:

$$\begin{array}{cccccccccc} \star & \dagger & 42 & 1 & 23 & 2 & 25 & 3 & \cdots \\ OQ & PA & OQ & PA & OQ & PA & OQ & PA & \end{array}$$

where:

- the first move is played by O and asks the result of **count**, given an empty context (so, $\star$ is a move representing the empty context)
- the second move is played by P and answers the initial question saying the result is a function (so, $\dagger$ is a move representing a function)

Games more formally

$[\![\vdash \textbf{count} : \text{int} \to \text{int}]\!]$ is a set of plays of the form:

$$\begin{array}{ccccccccc} \star & \dagger & 42 & 1 & 23 & 2 & 25 & 3 & \cdots \\ OQ & PA & OQ & PA & OQ & PA & OQ & PA & \end{array}$$

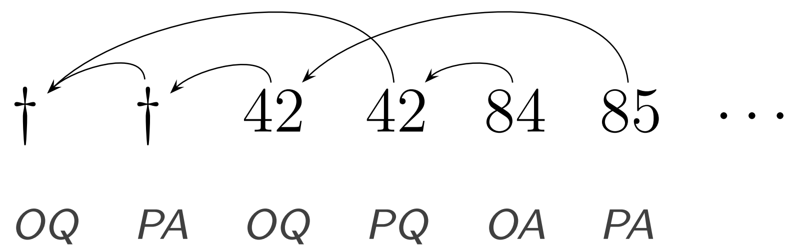
where:

- the first move is played by O and asks the result of **count**, given an empty context (so, $\star$ is a move representing the empty context)
- the second move is played by P and answers the initial question saying the result is a function (so, $\dagger$ is a move representing a function)
- from there on, we engage in a OQ-PA pattern:
  - O asks the result of the function on some input number
  - P answers by simply playing the number of times it has been called

Games more formally II

Recall **plusOne** $\equiv \lambda x^{\mathsf{int}}. \, f x + 1.$

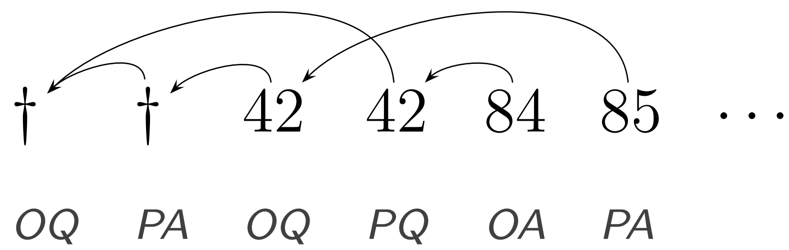
$[\![f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}]\!]$ is a set of plays of the form:



Games more formally II

Recall **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1.$

$[\![f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}]\!]$ is a set of plays of the form:



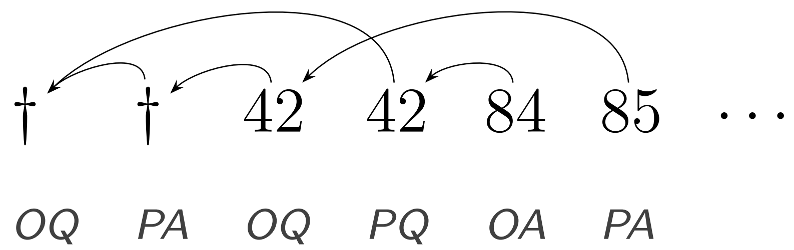
where, additionally to what we saw in $[\![$**count**$]\!]$,

- we also have **pointers** between moves
- each move has a pointer to an earlier move that **justifies** it

Games more formally II

Recall **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1.$

$[\![f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}]\!]$ is a set of plays of the form:



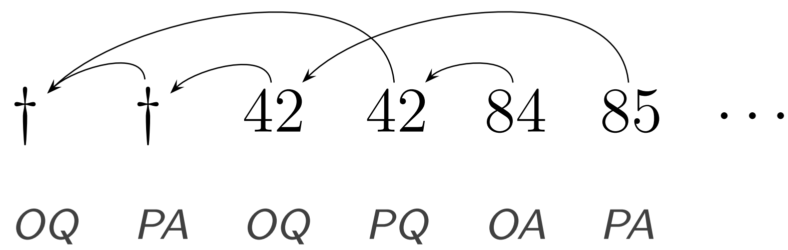
where, additionally to what we saw in $[\![$**count**$]\!]$,

- we also have **pointers** between moves
- each move has a pointer to an earlier move that **justifies** it
  - e.g. an answer points to its corresponding question

Games more formally II

Recall **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1.$

$[\![f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}]\!]$ is a set of plays of the form:



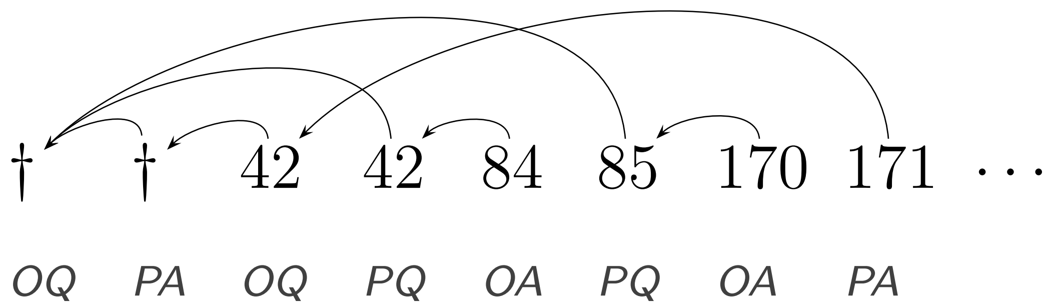
where, additionally to what we saw in $[\![$**count**$]\!]$,

- we also have **pointers** between moves
- each move has a pointer to an earlier move that **justifies** it
  - e.g. an answer points to its corresponding question
  - a question points to the † move that the question refers to
    - e.g. the 42 played by O is a question to second † (i.e. f')
    - whereas the 42 played by P is a question to first † (i.e. f)

Games more formally III

Recall **plusOne**^{2} $\equiv$ $\lambda x^{\mathsf{int}}.\, f(f x + 1) + 1$.

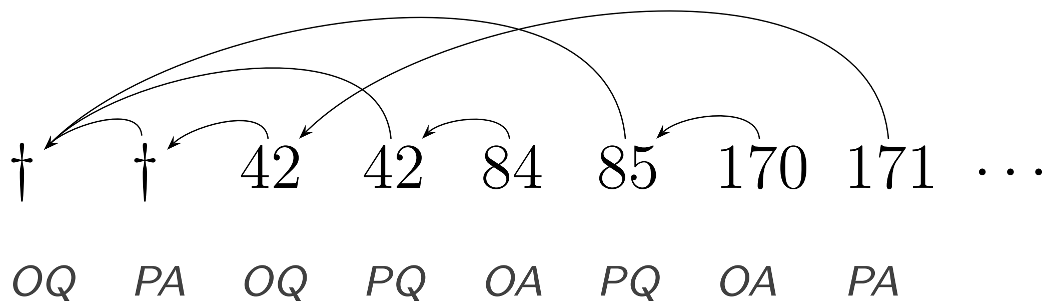
$[\![f : \mathsf{int} \to \mathsf{int} \vdash \textbf{plusOne}^2 : \mathsf{int} \to \mathsf{int}]\!]$ is a set of plays of the form:



Games more formally III

Recall **plusOne**$^2 \;\equiv\; \lambda x^{\mathsf{int}}.\, f(fx+1)+1$.

$[\![f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne**$^2 : \mathsf{int} \to \mathsf{int}]\!]$ is a set of plays of the form:



So, plays combine:

- the idea of executing a program and exchanging moves with its context
- with those moves potentially representing higher-order functions
  - the exchanged moves themselves enable more moves to be played
  - pointers are used to keep an order on what-is-played-where

Game duality

We know that P in a game is the modelled program. But, who is O?

- It is another program. But then, that program also has a game semantics!

Game duality

We know that P in a game is the modelled program. But, who is O?

- It is another program. But then, that program also has a game semantics!

$\vdash$ $\lambda y^{\mathsf{int}}. 2 * y : \mathsf{int} \to \mathsf{int}$

O what is the result?

Game duality

We know that P in a game is the modelled program. But, who is O?

- It is another program. But then, that program also has a game semantics!

$\vdash$ $\lambda y^{\mathsf{int}}. 2 * y : \mathsf{int} \to \mathsf{int}$

O what is the result?

P it is a function

Game duality

We know that P in a game is the modelled program. But, who is O?

- It is another program. But then, that program also has a game semantics!

$\vdash$	$\lambda y^{\mathsf{int}}. 2 * y : \mathsf{int} \to \mathsf{int}$
O	what is the result?
P	it is a function
O	what is the result for 42?

Game duality

We know that P in a game is the modelled program. But, who is O?

- It is another program. But then, that program also has a game semantics!

$\vdash$ $\lambda y^{\mathsf{int}}.\, 2 * y : \mathsf{int} \to \mathsf{int}$

O what is the result?

P it is a function

O what is the result for 42?

P it is 84

Game duality

We know that P in a game is the modelled program. But, who is O?

- It is another program. But then, that program also has a game semantics!

$\vdash \lambda y^{\mathsf{int}}. 2 * y : \mathsf{int} \to \mathsf{int}$		$f : \mathsf{int} \to \mathsf{int} \vdash \lambda x^{\mathsf{int}}. f x + 1 : \mathsf{int} \to \mathsf{int}$	
O	what is the result?	O	given function f, what is the result?
P	it is a function	P	it is a function
O	what is the result for 42?	O	what is the result for 42?
P	it is 84	P	what is the result of f for 42?
		O	it is 84
		P	it is 85

Game duality

We know that P in a game is the modelled program. But, who is O?

- It is another program. But then, that program also has a game semantics!

$\vdash \lambda y^{\mathsf{int}}.\, 2 * y : \mathsf{int} \to \mathsf{int}$		$f : \mathsf{int} \to \mathsf{int} \vdash \lambda x^{\mathsf{int}}.\, f x + 1 : \mathsf{int} \to \mathsf{int}$	
O	what is the result?		
P	it is a function	O	given function f, what is the result?
		P	it is a function
		O	what is the result for 42?
O	what is the result for 42?	P	what is the result of f for 42?
P	it is 84	O	it is 84
		P	it is 85

Game duality

If we write moves formally and re-arrange moves in space so that we see where they come from

- we see a **duality** between P and O in the middle component

	$\vdash \lambda y^{\mathsf{int}}. 2 * y$	: int $\to$ int		f : int $\to$ int	$\vdash$ $\lambda x^{\mathsf{int}}. f x + 1$: int $\to$ int
O	$\star$				
P		$\dagger$	O	$\dagger$	
			P		$\dagger$
			O		42
O		42	P	42	
P		84	O	84	
			P		85

Game duality

If we write moves formally and re-arrange moves in space so that we see where they come from

- we see a **duality** between P and O in the middle component

	$\vdash \lambda y^{\mathsf{int}}. 2 * y$	: int	→ int		f : int	→ int	$\vdash \lambda x^{\mathsf{int}}. f x + 1$: int	→ int
O	$\star$							
P			$\dagger$	O		$\dagger$		
				P				$\dagger$
				O			42	
O		42		P	42			
P			84	O		84		
				P				85

What is P on the LHS of the grey box, is O on the RHS, and viceversa.

Game duality and composition

If we write moves formally and re-arrange moves in space so that we see where they come from

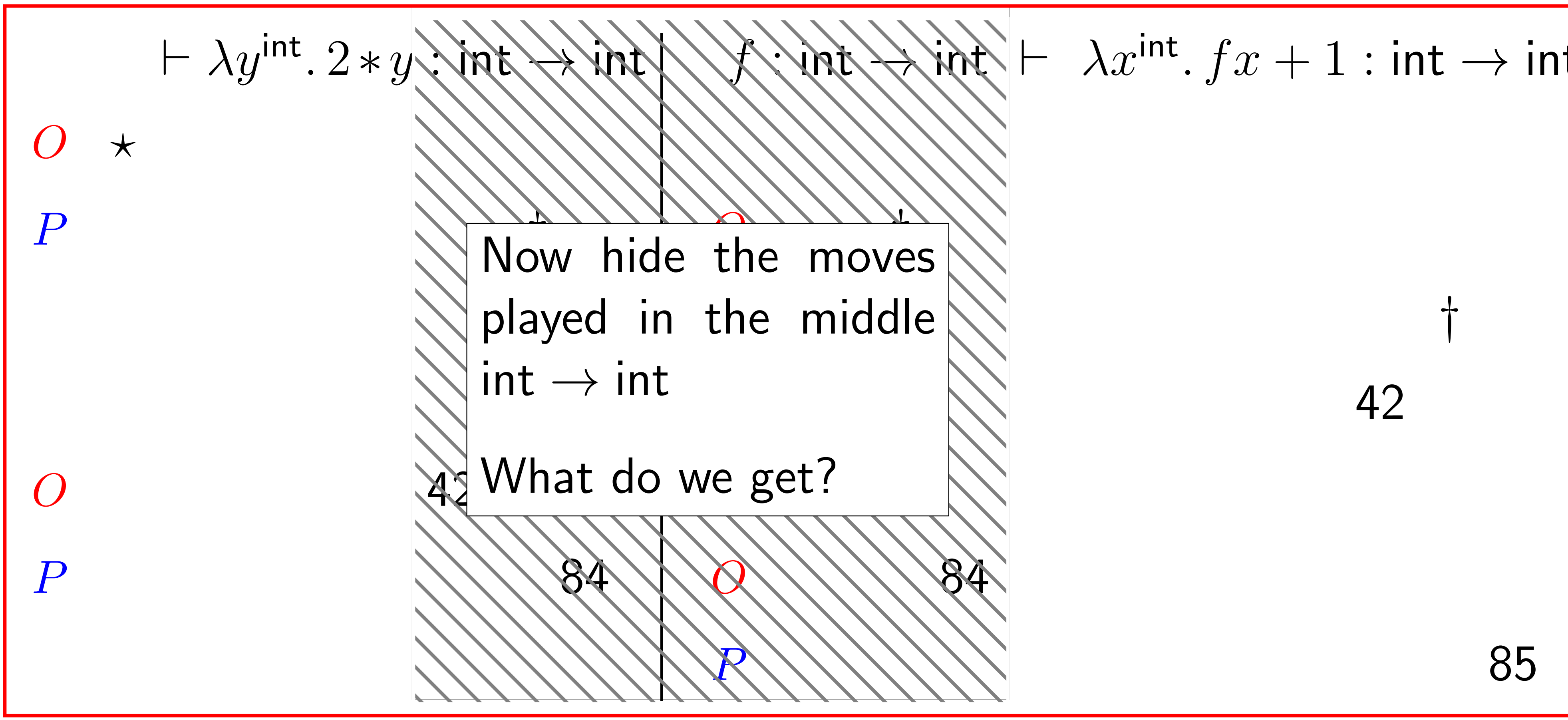
- we see a **duality** between P and O in the middle component

	$\vdash \lambda y^{\mathsf{int}}.\, 2 * y$:	int	$\to$ int		f : int	$\to$ int	$\vdash \lambda x^{\mathsf{int}}.\, f x + 1$: int	$\to$ int
O	$\star$							
P			$\dagger$	O		$\dagger$		
				P				$\dagger$
				O			42	
O		42		P	42			
P			84	O		84		
				P				85

Game duality and composition

If we write moves formally and re-arrange moves in space so that we see where they come from

- we see a **duality** between P and O in the middle component



Game duality and composition

	$\vdash ??? : \mathsf{int} \to \mathsf{int}$
O	$\star$
P	$\dagger$
O	42
P	85

Game duality and composition

$\vdash ??? : \mathsf{int} \to \mathsf{int}$

O $\star$

P $\dagger$

O 42

P 85

We get the game corresponding to the **composition** of the two programs:

$$\vdash \ \mathsf{let}\,(f = \lambda y^{\mathsf{int}}.\, 2 * y)\,\mathsf{in}\,\lambda x^{\mathsf{int}}.\, f x + 1$$

i.e. $\vdash \ \lambda x^{\mathsf{int}}.\, 2 * x + 1$.

So, we can compose games with a common right/left component:

- synchronising moves in common component (using duality)
- and hiding those moves.

This is analogous to how functions compose.

More plays

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$, where **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1$.

The corresponding game has plays for **every possible behaviour** of f:

- given function f, what is the result of **plusOne**?
- it is a function f'

More plays

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$, where **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1$.

The corresponding game has plays for **every possible behaviour** of f:

- given function f, what is the result of **plusOne**?
- it is a function f'
  - what is the result of f' on integer i?

More plays

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$, where **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1$.

The corresponding game has plays for **every possible behaviour** of f:

- given function f, what is the result of **plusOne**?
- it is a function f'
  - what is the result of f' on integer i ?
    - what is the result of f on i ?

More plays

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$, where **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1$.

The corresponding game has plays for **every possible behaviour** of f:

- ○ given function f, what is the result of **plusOne**?
- • it is a function f'
  - ○ what is the result of f' on integer i ?
    - • what is the result of f on i ?
    - ○ it is j

More plays

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** : $\mathsf{int} \to \mathsf{int}$, where **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1$.

The corresponding game has plays for **every possible behaviour** of f:

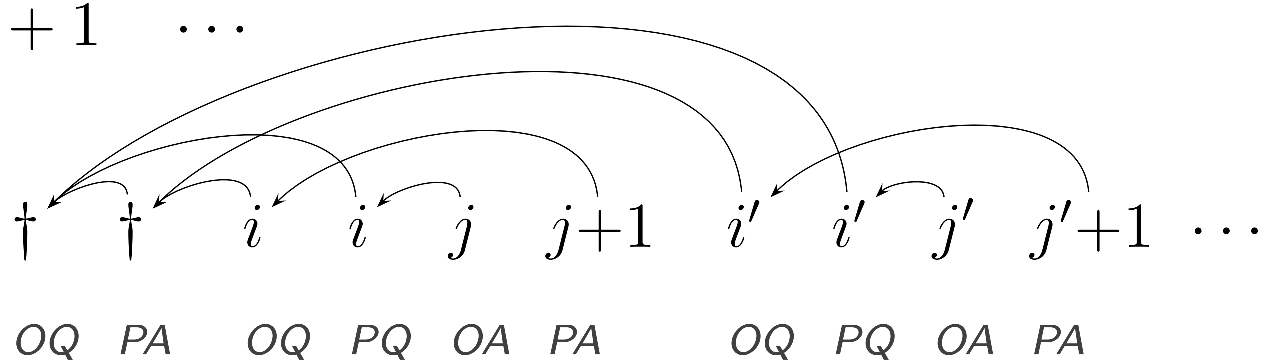
- ○ given function f, what is the result of **plusOne**?
- • it is a function f'
  - ○ what is the result of f' on integer i ?
    - • what is the result of f on i ?
    - ○ it is j
  - • it is $j + 1$ $\quad\cdots$

More plays

Example. Consider a program $f : \mathsf{int} \to \mathsf{int} \vdash$ **plusOne** $: \mathsf{int} \to \mathsf{int}$, where **plusOne** $\equiv \lambda x^{\mathsf{int}}.\, f x + 1$.

The corresponding game has plays for **every possible behaviour** of f:

- ○ given function f, what is the result of **plusOne**?
- • it is a function f'
  - ○ what is the result of f' on integer i ?
    - • what is the result of f on i ?
    - ○ it is j
  - • it is $j + 1$ $\cdots$



$[\![$**plusOne**$]\!]$ contains **all plays** of that form (i.e. for all $i, j, i', j', \dots \in \mathbb{Z}$)

Results

Games model programs under any possible context:

- they contain plays for every O move allowed
- they include conditions that disallow spurious plays.

And this is the key to full-abstraction results:

$$M \cong N \iff [\![M]\!] = [\![N]\!]$$

Results

Games model programs under any possible context:

- they contain plays for every O move allowed
- they include conditions that disallow spurious plays.

And this is the key to full-abstraction results:

$$M \cong N \iff [\![M]\!] = [\![N]\!]$$

After the original papers on PCF, there was a series of works covering extensions of PCF with effects: local state, local higher-order state, non-determinism, probabilities, control operators, etc.

More recently, games have been extended to languages with data-generating effects, like references, objects, channels, etc. Nowadays, games capture a wide-range of higher-order languages, typically fragments of OCaML and Java.

See the Lecture Notes for references.

Exercises

1. Consider the following alternative notion of equivalence for PCF terms:

   Given $\Gamma \vdash M_1 : \theta$ and $\Gamma \vdash M_2 : \theta$, we let $M_1 \cong' M_2$ if, for every context C such that $\vdash C[M_1] : \mathsf{int}$ and $\vdash C[M_2] : \mathsf{int}$, and for all $i \in \mathbb{Z}$, $C[M_1] \longrightarrow^* i \iff C[M_2] \longrightarrow^* i$.

   Prove that $\cong'$ coincides with $\cong$.

2. We use the following shorthand notation: $\mathsf{let}\, x = M \,\mathsf{in}\, N \;\equiv\; (\lambda x.N)M$.

   Using the operational semantics of PCF:

   - Verify that $(\mathsf{let}\,(f = \lambda y^{\mathsf{int}}.\, 2 * y)\,\mathsf{in}\,\lambda x^{\mathsf{int}}.\, fx + 1)z \;\longrightarrow^*\; 2 * z + 1$.
   - Compute $(\mathsf{let}\,(f = \lambda y^{\mathsf{int}}.\, 2 * y)\,\mathsf{in}\,\lambda x^{\mathsf{int}}.\, f(fx + 1) + 1)\, z$.

3. Working as in Slides 24-25, compose game-semantically $\vdash \lambda y^{\mathsf{int}}.\, 2 * y : \mathsf{int} \to \mathsf{int}$ and $f : \mathsf{int} \to \mathsf{int} \vdash \lambda x^{\mathsf{int}}.\, f(fx + 1) + 1 : \mathsf{int} \to \mathsf{int}$.

4. Consider the program

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

   obtained by λ-abstracting **plusOne**. Working first informally (using dialogues) and then formally (using plays), express the plays in $[\![\lambda f^{\mathsf{int}\to\mathsf{int}}.\lambda x^{\mathsf{int}}.\, fx + 1]\!]$:

   - First, assuming that O plays a question † (for f) only once.
   - Consider how the plays would look like if O plays the † more than once.