

# *A Concrete Presentation of Game Semantics*

William Blum

Joint work with C.-H. Luke Ong

School of Informatics, University of Edinburgh – Oxford University Computing  
Laboratory

Galop, 5 April 2008

# Overview

- ▶ Game-semantic models are **abstract** *i.e.* independent of the syntax of the denoted term. We give here a **concrete** *i.e.* syntactic representation of game semantics where:
  - ▶ The arena game is ‘incarnated’ by some abstract syntax tree of the term,
  - ▶ Uncovered plays are given by traversals over this tree.
- ▶ A “Correspondence Theorem” establishes the relationship between the game-semantic and traversal models.
- ▶ The tool HOG illustrates this correspondence.
- ▶ Example of application: computing infinite trees generated by *higher-order recursion schemes*.

# Outline

## The correspondence

- Game semantics

- Computation tree

- The Correspondence Theorem

- Example

- Composition

- Demo

## Applications

- Higher-order grammars

- Other applications

## Conclusion & Future Works

# Outline

## The correspondence

- Game semantics

- Computation tree

- The Correspondence Theorem

- Example

- Composition

- Demo

## Applications

- Higher-order grammars

- Other applications

## Conclusion & Future Works

## Game semantics

Model of programming languages based on games (Abramsky et al.; Hyland and Ong; Nickau)

- ▶ 2 players: **O**pponent (system) and **P**roponent (program)
- ▶ The term type induces an **arena** defining the possible moves

$$\llbracket \mathbb{N} \rrbracket = \begin{array}{c} q \\ / \quad | \quad \backslash \\ 0 \quad 1 \quad \dots \end{array} \qquad \llbracket \mathbb{N} \rightarrow \mathbb{N} \rrbracket = \begin{array}{c} q^0 \\ / \quad | \quad \backslash \\ q^1 \quad 0 \quad 1 \quad \dots \\ / \quad | \quad \backslash \\ 0 \quad 1 \quad \dots \end{array}$$

- ▶ **Play** = sequence of moves played alternatively by O and P with justification pointers.
- ▶ **Strategy for P** = prefix-closed set of plays.  $sab$  in the strategy means that P should respond  $b$  when O plays  $a$  in position  $s$ .
- ▶ The **denotation** of a term  $M$ , written  $\llbracket M \rrbracket$ , is a strategy for P.
- ▶  $\llbracket 7 : \mathbb{N} \rrbracket = \{\epsilon, q, q 7\}$   
 $\llbracket \text{succ} : \mathbb{N} \rightarrow \mathbb{N} \rrbracket = \text{Pref}(\{q^0 q^1 n(n+1) \mid n \in \mathbb{N}\})$
- ▶ **Compositionality**:  $\llbracket \text{succ } 7 \rrbracket = \llbracket \text{succ} \rrbracket; \llbracket 7 \rrbracket$

## Game semantics: composition

- ▶ Composition is done by CSP-composition + hiding: If  $\sigma : A \rightarrow B$  and  $\mu : B \rightarrow C$  then

$$“ \sigma ; \mu = (\sigma \parallel \mu) \upharpoonright A, C ”$$

- ▶ The **fully revealed** game denotation, written  $\langle\langle M \rangle\rangle$ , denotes the set of plays obtained by not performing hiding of internal moves during composition.

## Game semantics: composition

- ▶ Composition is done by CSP-composition + hiding: If  $\sigma : A \rightarrow B$  and  $\mu : B \rightarrow C$  then

$$“ \sigma ; \mu = (\sigma \parallel \mu) \upharpoonright A, C ”$$

- ▶ The **fully revealed** game denotation, written  $\langle\langle M \rangle\rangle$ , denotes the set of plays obtained by not performing hiding of internal moves during composition.

## Computation tree

We fix a simply-typed term  $\Gamma \vdash M : T$ .

*Computation tree* of  $M$  is the AST of its  $\eta$ -long normal form.

- ▶ The  $\eta$ -expansion of  $M : A \rightarrow B$  is  $\lambda x : A. Mx : A \rightarrow B$ .
- ▶ The  $\eta$ -long normal form of  $M$  is obtained by hereditarily  $\eta$ -expanding every subterm of  $M$  occurring at an operand position or as the body of a  $\lambda$ -abstraction.

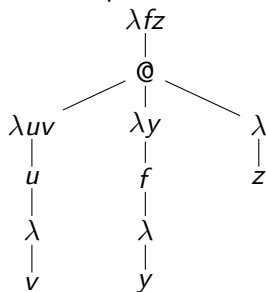
Example:

$$\vdash \lambda f^{o \rightarrow o}. (\lambda u^{o \rightarrow o}. u) f : (o \rightarrow o) \rightarrow o \rightarrow o$$

Its  $\eta$ -long normal form is

$$\begin{aligned} &\vdash \lambda f^{o \rightarrow o} z^o. \\ &\quad (\lambda u^{o \rightarrow o} v^o. u(\lambda. v)) \\ &\quad (\lambda y^o. f y) \\ &\quad (\lambda. z) \\ &: (o \rightarrow o) \rightarrow o \rightarrow o \end{aligned}$$

The computation tree is:






## Justified sequence

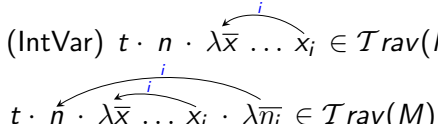
- ▶ We define an **enabling relation**  $\vdash$  on the set of nodes:
  - ▶ a bound variable is enabled by its binder;
  - ▶ a free variable is enabled by the root  $\ast$ ;
  - ▶ a lambda node is enabled by its parent node;
  - ▶ an @-node has no enabler.
- ▶ Distinction between external nodes  $N^{\ast\vdash}$  (hereditarily justified by the root) and the internal nodes  $N^{\text{@}\vdash}$  (her. just. by an @-node).
- ▶ A **justified sequence** is a sequence of nodes such that all the non @-nodes have a justification pointer respecting the relation  $\vdash$ .
- ▶ The analogy with game semantics is:
  - ▶  $\lambda$ -nodes  $\equiv$  O-moves
  - ▶ @-nodes and variable-nodes  $\equiv$  P-moves

We can define notions of P-view and O-view, alternation, P-visibility, O-visibility.

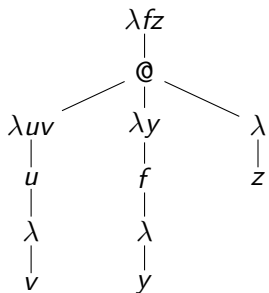
## Traversals rules

The computation is described by a set  $\mathcal{T}rav(M)$  of justified sequences called **traversals** and given by induction over the rules:

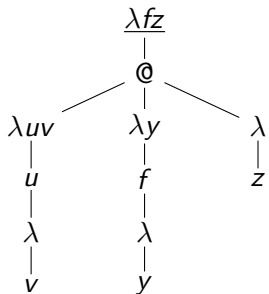
- ▶ (Empty)  $\epsilon \in \mathcal{T}rav(M)$
- ▶ (Root)  $\ast \in \mathcal{T}rav(M)$
- ▶ (Lam)  $t \cdot \lambda \bar{\xi} \in \mathcal{T}rav(M) \implies t \cdot \lambda \bar{\xi} \cdot n \in \mathcal{T}rav(M)$  where  $n$  is  $\lambda \bar{\xi}$ 's child and is justified by the only occurrence of its enabler in the P-view
- ▶ (App)  $t \cdot @ \in \mathcal{T}rav(M) \implies t \cdot @ \cdot n \in \mathcal{T}rav(M)$ .  

- ▶ (ExtVar)  $t \cdot x \in \mathcal{T}rav(M)$ ,  $x \in N_{\text{var}}^{\circledast \vdash} \implies t \cdot x \cdot n \in \mathcal{T}rav(M)$  for any  $\lambda$ -node  $n$  justified by some occurrence of its parent node in  $\llcorner t \lrcorner$ .

- ▶ (IntVar)  $t \cdot n \cdot \lambda \bar{x} \dots x_i \in \mathcal{T}rav(M)$ ,  $x_i \in N_{\text{var}}^{\circledast \vdash} \implies$   
 $t \cdot \bar{n} \cdot \lambda \bar{x} \dots x_i \cdot \lambda \bar{\eta}_i \in \mathcal{T}rav(M)$ .

## Example of traversal

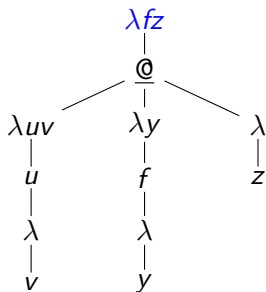


## Example of traversal



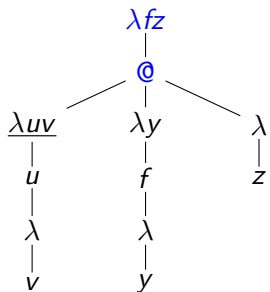
$$t = \lambda fz$$

## Example of traversal



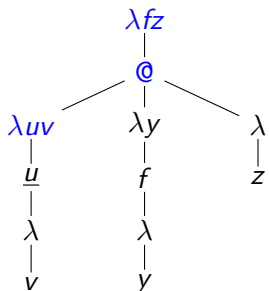
$$t = \lambda fz \cdot @$$

## Example of traversal



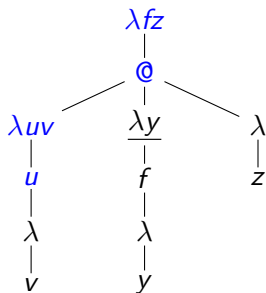
$$t = \lambda f z \cdot @ \cdot \lambda u v$$

## Example of traversal



$$t = \lambda fz \cdot @ \cdot \lambda uv \cdot u$$

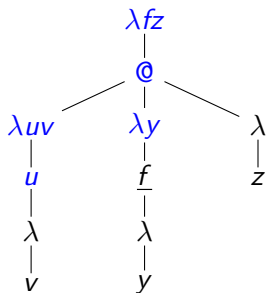
## Example of traversal



$$t = \lambda f z \cdot @ \cdot \lambda u v \cdot u \cdot \lambda y$$

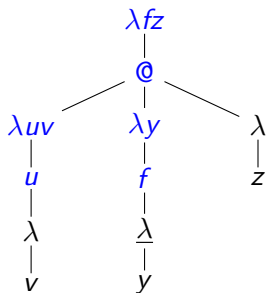


## Example of traversal



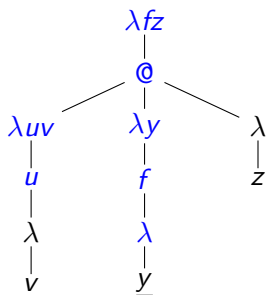
$t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f$

## Example of traversal



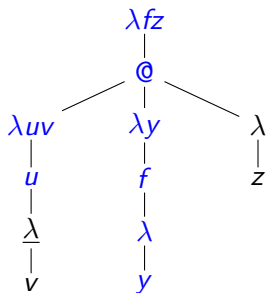
$t = \lambda f z \cdot @ \cdot \lambda u v \cdot u \cdot \lambda y \cdot f \cdot \lambda$

## Example of traversal



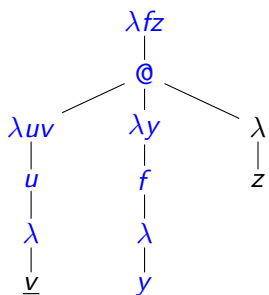
$t = \lambda f z \cdot @ \cdot \lambda u v \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y$

## Example of traversal



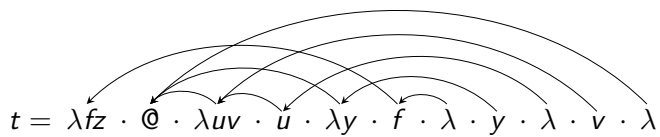
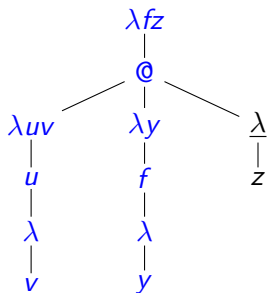
$t = \lambda f z \cdot @ \cdot \lambda u v \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y \cdot \lambda$

## Example of traversal

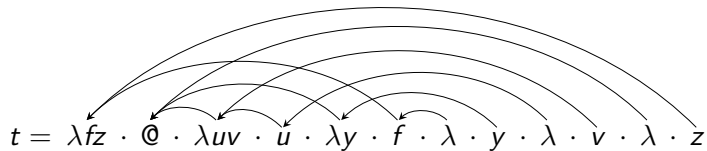
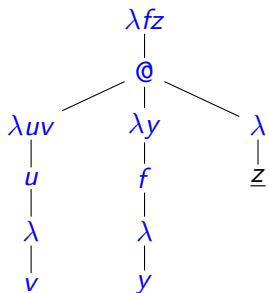


$t = \lambda f z \cdot @ \cdot \lambda u v \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y \cdot \lambda \cdot v$

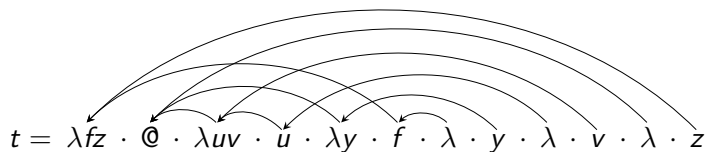
## Example of traversal



## Example of traversal



## Operations on traversals



- ▶ The **reduction of a traversal** is obtained by keeping only the occurrences hereditarily justified by the root:

$$t \upharpoonright \lambda fz = \lambda fz \quad f \quad \lambda \quad z$$

The diagram shows the reduced traversal  $t \upharpoonright \lambda fz = \lambda fz \quad f \quad \lambda \quad z$ . Curved arrows indicate hereditary justification from the root  $\lambda fz$  to its children  $f$ ,  $\lambda$ , and  $z$ .

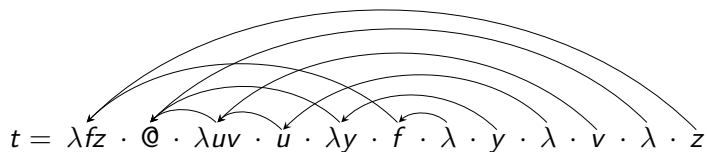
- ▶ *@-nodes removal:*

$$t - @ = \lambda fz \quad \lambda uv \quad u \quad \lambda y \quad f \quad \lambda \quad y \quad \lambda \quad v \quad \lambda \quad z$$

The diagram shows the traversal  $t - @ = \lambda fz \quad \lambda uv \quad u \quad \lambda y \quad f \quad \lambda \quad y \quad \lambda \quad v \quad \lambda \quad z$ . Curved arrows indicate hereditary justification from the root  $\lambda fz$  to its children  $\lambda uv$ ,  $u$ ,  $\lambda y$ ,  $f$ ,  $\lambda$ , and  $y$ . Additional arrows show justification from  $\lambda uv$  to  $u$  and  $v$ , and from  $\lambda y$  to  $y$ .



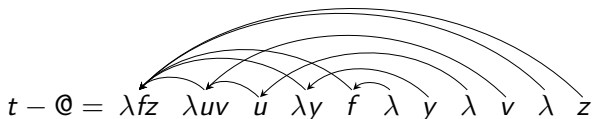
## Operations on traversals



- ▶ The **reduction of a traversal** is obtained by keeping only the occurrences hereditarily justified by the root:

$$t \upharpoonright \lambda f z = \lambda f z \quad \overset{\curvearrowright}{\underset{\curvearrowleft}{f}} \quad \lambda \quad z$$

- ▶ *@-nodes removal:*



# The Correspondence Theorem

Let  $M$  be a simply typed term of type  $T$ . There exists a function  $\varphi$  from the nodes of the **computation tree** to the moves of the **arenas** of  $\langle\langle T \rangle\rangle$  such that

$$\varphi : \mathcal{T}rav(M)^{-\textcircled{c}} \xrightarrow{\cong} \langle\langle M \rangle\rangle$$

$$\varphi : \mathcal{T}rav(M)^{\uparrow\textcircled{*}} \xrightarrow{\cong} \llbracket M \rrbracket .$$

where

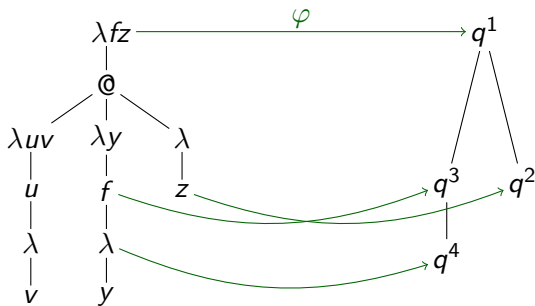
- ▶  $\mathcal{T}rav(M)$  = set of traversals of the computation tree of  $M$
- ▶  $\mathcal{T}rav(M)^{\uparrow\textcircled{*}} = \{t \upharpoonright t_0 \mid t \in \mathcal{T}rav(M)\}$
- ▶  $\mathcal{T}rav(M)^{-\textcircled{c}} = \{t - \textcircled{c} \mid t \in \mathcal{T}rav(M)\}$
- ▶  $\llbracket M \rrbracket$  = game-semantic denotation of  $M$
- ▶  $\langle\langle M \rangle\rangle$  = revealed denotation of  $M$ .

## More correspondences

Computation tree notions	Game-semantic equivalents
computation tree	arena(s)
traversal	uncovered play
reduced traversal	play
path in the computation tree	P-view of an uncovered play

Example:  $\vdash \lambda f^{o \rightarrow o} . (\lambda u^{o \rightarrow o} . u) f : (o \rightarrow o) \rightarrow o \rightarrow o$

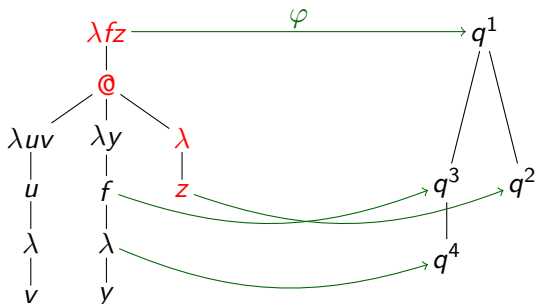
Left: computation tree. Right: arena.



- ▶  $t = \lambda f z \cdot @ \cdot \lambda u v \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y \cdot \lambda \cdot v \cdot \lambda \cdot z$
- ▶  $\lceil t \rceil = \lambda f z \cdot @ \cdot \lambda \cdot z$
- ▶  $\varphi(t \upharpoonright \lambda f z) = \varphi(\lambda f z \cdot f \cdot \lambda \cdot z) = q^1 q^3 q^4 q^2 \in \llbracket M \rrbracket$ .

Example:  $\vdash \lambda f^{o \rightarrow o}. (\lambda u^{o \rightarrow o}. u) f : (o \rightarrow o) \rightarrow o \rightarrow o$

Left: computation tree. Right: arena.



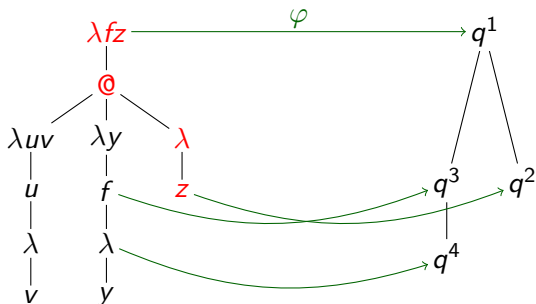
▶  $t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y \cdot \lambda \cdot v \cdot \lambda \cdot z$

▶  $\ulcorner t \urcorner = \lambda fz \cdot @ \cdot \lambda \cdot z$

▶  $\varphi(t \upharpoonright \lambda fz) = \varphi(\lambda fz \cdot f \cdot \lambda \cdot z) = q^1 q^3 q^4 q^2 \in \llbracket M \rrbracket$ .

Example:  $\vdash \lambda f^{o \rightarrow o} . (\lambda u^{o \rightarrow o} . u) f : (o \rightarrow o) \rightarrow o \rightarrow o$

Left: computation tree. Right: arena.



▶  $t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y \cdot \lambda \cdot v \cdot \lambda \cdot z$

▶  $\ulcorner t \urcorner = \lambda fz \cdot @ \cdot \lambda \cdot z$

▶  $\varphi(t \upharpoonright \lambda fz) = \varphi(\lambda fz \cdot f \cdot \lambda \cdot z) = q^1 q^3 q^4 q^2 \in \llbracket M \rrbracket$ .

# Composition

$$\llbracket M \rrbracket; \llbracket N \rrbracket = \llbracket \lambda x. N(Mx) \rrbracket$$

Example:

▶  $A = o$ ,  $B = o \rightarrow o$ ,  $C = ((o \rightarrow o) \rightarrow o) \rightarrow o$

▶  $\sigma = \llbracket \lambda x^o v^o. x : A \rightarrow B \rrbracket$

▶  $\mu = \llbracket \lambda y^B \varphi^{((o \rightarrow o) \rightarrow o)}. \varphi(\lambda u^o. y(\lambda. u)) : B \rightarrow C \rrbracket$

We then have  $\sigma ; \mu = \llbracket \lambda x \varphi. \varphi(\lambda u. x) : A \rightarrow C \rrbracket$ .

$$o \xrightarrow{\sigma} (o \rightarrow o) \xrightarrow{\mu} (((o \rightarrow o) \rightarrow o) \rightarrow o)$$

# Composition

$$\llbracket M \rrbracket; \llbracket N \rrbracket = \llbracket \lambda x. N(Mx) \rrbracket$$

Example:

▶  $A = o, B = o \rightarrow o, C = ((o \rightarrow o) \rightarrow o) \rightarrow o$

▶  $\sigma = \llbracket \lambda x^o v^o. x : A \rightarrow B \rrbracket$

▶  $\mu = \llbracket \lambda y^B \varphi^{((o \rightarrow o) \rightarrow o)}. \varphi(\lambda u^o. y(\lambda. u)) : B \rightarrow C \rrbracket$

We then have  $\sigma ; \mu = \llbracket \lambda x \varphi. \varphi(\lambda u. x) : A \rightarrow C \rrbracket$ .

$$o \xrightarrow{\sigma} (o \rightarrow o) \xrightarrow{\mu} (((o \rightarrow o) \rightarrow o) \rightarrow o)$$

$\lambda x \varphi \bullet \lambda y \varphi$



# Composition

$$\llbracket M \rrbracket; \llbracket N \rrbracket = \llbracket \lambda x. N(Mx) \rrbracket$$

Example:

▶  $A = o$ ,  $B = o \rightarrow o$ ,  $C = ((o \rightarrow o) \rightarrow o) \rightarrow o$

▶  $\sigma = \llbracket \lambda x^o v^o. x : A \rightarrow B \rrbracket$

▶  $\mu = \llbracket \lambda y^B \varphi^{((o \rightarrow o) \rightarrow o)}. \varphi(\lambda u^o. y(\lambda. u)) : B \rightarrow C \rrbracket$

We then have  $\sigma ; \mu = \llbracket \lambda x \varphi. \varphi(\lambda u. x) : A \rightarrow C \rrbracket$ .

$$o \xrightarrow{\sigma} (o \rightarrow o) \xrightarrow{\mu} (((o \rightarrow o) \rightarrow o) \rightarrow o)$$

$$\begin{array}{c} \curvearrowright \lambda x \varphi \bullet \lambda y \varphi \\ \varphi \circ \varphi \end{array}$$

# Composition

$$\llbracket M \rrbracket; \llbracket N \rrbracket = \llbracket \lambda x. N(Mx) \rrbracket$$

Example:

▶  $A = o$ ,  $B = o \rightarrow o$ ,  $C = ((o \rightarrow o) \rightarrow o) \rightarrow o$

▶  $\sigma = \llbracket \lambda x^o v^o. x : A \rightarrow B \rrbracket$

▶  $\mu = \llbracket \lambda y^B \varphi^{((o \rightarrow o) \rightarrow o)}. \varphi(\lambda u^o. y(\lambda. u)) : B \rightarrow C \rrbracket$

We then have  $\sigma \circ \mu = \llbracket \lambda x \varphi. \varphi(\lambda u. x) : A \rightarrow C \rrbracket$ .

$$o \xrightarrow{\sigma} (o \rightarrow o) \xrightarrow{\mu} (((o \rightarrow o) \rightarrow o) \rightarrow o)$$

$$\lambda u \bullet \lambda u \quad \varphi \circ \varphi \quad \lambda x \varphi \bullet \lambda y \varphi$$

# Composition

$$\llbracket M \rrbracket; \llbracket N \rrbracket = \llbracket \lambda x. N(Mx) \rrbracket$$

Example:

▶  $A = o, B = o \rightarrow o, C = ((o \rightarrow o) \rightarrow o) \rightarrow o$

▶  $\sigma = \llbracket \lambda x^o v^o. x : A \rightarrow B \rrbracket$

▶  $\mu = \llbracket \lambda y^B \varphi^{((o \rightarrow o) \rightarrow o)}. \varphi(\lambda u^o. y(\lambda. u)) : B \rightarrow C \rrbracket$

We then have  $\sigma ; \mu = \llbracket \lambda x \varphi. \varphi(\lambda u. x) : A \rightarrow C \rrbracket$ .

$$o \xrightarrow{\sigma} (o \rightarrow o) \xrightarrow{\mu} (((o \rightarrow o) \rightarrow o) \rightarrow o)$$

$\lambda x v \bullet y$   $\xrightarrow{\sigma}$   $(o \rightarrow o)$   $\xrightarrow{\mu}$   $(((o \rightarrow o) \rightarrow o) \rightarrow o)$

$\lambda x \varphi \bullet \lambda y \varphi$

$\lambda u \bullet \lambda u$   $\xrightarrow{\varphi \circ \varphi}$

# Composition

$$\llbracket M \rrbracket; \llbracket N \rrbracket = \llbracket \lambda x. N(Mx) \rrbracket$$

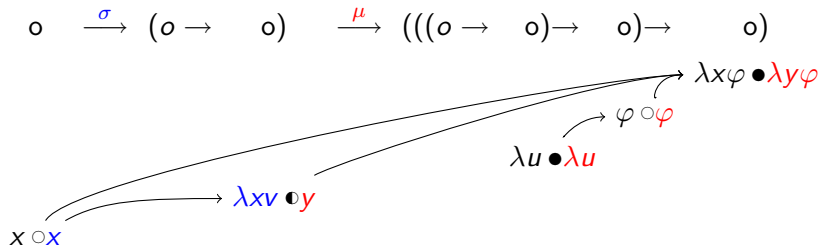
Example:

▶  $A = o, B = o \rightarrow o, C = ((o \rightarrow o) \rightarrow o) \rightarrow o$

▶  $\sigma = \llbracket \lambda x^o v^o. x : A \rightarrow B \rrbracket$

▶  $\mu = \llbracket \lambda y^B \varphi^{((o \rightarrow o) \rightarrow o)}. \varphi(\lambda u^o. y(\lambda. u)) : B \rightarrow C \rrbracket$

We then have  $\sigma \circ \mu = \llbracket \lambda x \varphi. \varphi(\lambda u. x) : A \rightarrow C \rrbracket$ .



# Tool demo

HOG Version 0.0.9 - [Traversal calculator]

File Window Help

Worksheet Node operations Sequence operations

Import... Save... + Edit label <

New Q-View Duplicate Star Hereditary projection

Delete P-View Prefix Extension Subterm projection

Latex export Traversal game

Sequence... Worksheet... New Undo

Graph... Pick a node in the graph!

Console:

- Opening urzyczyn.rs
- Opening example.lmd
- Opening term2.lmd

# Benefits

- ▶ **Pedagogical:** Game semantics is sometimes considered hard to understand. Partly because of some obscure technical definitions.
  - ▶ A **P-view** is just a **control point** in the program AST. The **O-view** is the dual *i.e.* the control point of the environment;
  - ▶ **Innocence** means that the current control point determines the next action taken by the program.
  - ▶ Adding reference variables breaks innocence because of side-effects.
  - ▶ **Visibility** restricts the program to access only code that is in scope.
  - ▶ Adding general reference breaks visibility: *e.g.*  
 $\text{new } x := \lambda y.y \text{ in } x \ a;$
- ▶ **Efficient:** top-down computation of the game denotation as opposed to a compositional bottom-up approach.
  - ▶ only the relevant O-moves of the subterms are considered;
  - ▶ hiding performed only once at the end;
  - ▶ composition can be done at the syntactic level;
  - ▶ traversals ending with an internal move have an O-view of length  $\mathcal{O}(\text{ord } M)$ .

# Outline

## The correspondence

Game semantics

Computation tree

The Correspondence Theorem

Example

Composition

Demo

## Applications

Higher-order grammars

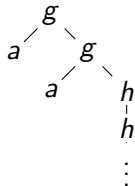
Other applications

## Conclusion & Future Works

# Higher-order grammars

- ▶ A **higher-order grammar** is formally given by a tuple  $\langle \Sigma, \mathcal{N}, \mathcal{R}, \mathcal{S} \rangle$  (terminals, non-terminals, rewriting rules, starting symbol)
- ▶ Higher-order grammars used as generators of word languages or trees are called **recursion schemes** (Maslov, 1974).
- ▶ Example of a tree-generating order-2 recursion scheme:

$$\begin{aligned} S &\rightarrow H a \\ H z^o &\rightarrow F(g z) \\ F \phi^o \rightarrow o &\rightarrow \phi(\phi(F h)) \end{aligned}$$



Terminals:  $a : o$  and  $g, h : o \rightarrow o$ . Non-terminals:  $S : o$ ,  $H : o \rightarrow o$  and  $F : (o \rightarrow o) \rightarrow o$ .



## Tree generating Higher-Order Pushdown Automata

HOPDAs generalize PDAs to nested stacks: a 1-PDA is a standard PDA; an order  $n$ -HOPDA manipulates a stack with  $n$  levels of nesting.

Formally given by a tuple  $\langle Q, \Sigma, \Gamma, \delta, q_o, \perp \rangle$  where

$\delta \subseteq Q \times \Gamma \rightarrow Q \times \text{Op}_n$  and the operations of  $\text{Op}_n$  are:

- ▶  $push_1 a$ : pushes the element  $a$  on the top level 1-stack;
- ▶  $pop_1$ : pops the top element of the top level 1-stack;
- ▶  $push_j, j > 1$ : duplicates the top level  $j - 1$ -stack;
- ▶  $pop_j, j > 1$ : deletes the top level  $j - 1$ -stack.

Additionally, we have an operation for generating tree-node:

- ▶  $emit_f(q_1, \dots, q_k)$ , for each node-constructor  $f : o^k \rightarrow o \in \Sigma$ .

### Theorem

*For generating word or tree languages, order- $n$  pushdown automata are equivalent to order- $n$  **safe** recursion scheme [Damm82, KNU02].*

What is the automata equivalent of unrestricted recursion schemes?

# Collapsible Pushdown Automaton (CPDA)

- ▶ Hague, Murawski, Ong, Serre, LICS08:
  - ▶ it is an extension of HOPDA where each symbol in the stack can have a pointer to a sub-stack occurring underneath it;
  - ▶ it can “collapse” the stack by following the pointer associated to the top element.
- ▶ There is a transformation from HORS to CPDA and conversely.
- ▶ The first transformation is based on the traversal theory:
  1. compute the computation graph of the HORS by taking the  $\eta$ -long nf of each grammar rule;
  2. construct a CPDA “simulating” the HORS by calculating the traversals of its computation graph. This is possible because the computed term is of ground type.
- ▶ Tool demo...

# Collapsible Pushdown Automaton (CPDA)

- ▶ Hague, Murawski, Ong, Serre, LICS08:
  - ▶ it is an extension of HOPDA where each symbol in the stack can have a pointer to a sub-stack occurring underneath it;
  - ▶ it can “collapse” the stack by following the pointer associated to the top element.
- ▶ There is a transformation from HORS to CPDA and conversely.
- ▶ The first transformation is based on the traversal theory:
  1. compute the computation graph of the HORS by taking the  $\eta$ -long nf of each grammar rule;
  2. construct a CPDA “simulating” the HORS by calculating the traversals of its computation graph. This is possible because the computed term is of ground type.
- ▶ Tool demo...

## Other applications, related works

- ▶ **Verification:** Knapik *et. al.* (2002) showed that **MSO model checking** for trees generated by HORS of any order and verifying the **safety restriction** (a syntactic restriction that constrains the occurrences of variables according to their orders) is decidable. Using the notions of computation tree/traversal Ong was able to show (LICS06) that this result still holds in the unrestricted case.
- ▶ Studying the effect of syntactic restrictions on the game semantics model. e.g. One can show that **pointers are uniquely recoverable** in the game denotation of terms satisfying the safety restriction.

### Related works:

- ▶ Stirling recently proved decidability of higher-order pattern matching with a game-semantic approach relying on equivalent notions of computation tree and traversal.

## Other applications, related works

- ▶ **Verification:** Knapik *et. al.* (2002) showed that **MSO model checking** for trees generated by HORS of any order and verifying the **safety restriction** (a syntactic restriction that constrains the occurrences of variables according to their orders) is decidable. Using the notions of computation tree/traversal Ong was able to show (LICS06) that this result still holds in the unrestricted case.
- ▶ Studying the effect of syntactic restrictions on the game semantics model. e.g. One can show that **pointers are uniquely recoverable** in the game denotation of terms satisfying the safety restriction.

### Related works:

- ▶ Stirling recently proved decidability of higher-order pattern matching with a game-semantic approach relying on equivalent notions of computation tree and traversal.

## Other applications, related works

- ▶ **Verification:** Knapik *et. al.* (2002) showed that **MSO model checking** for trees generated by HORS of any order and verifying the **safety restriction** (a syntactic restriction that constrains the occurrences of variables according to their orders) is decidable. Using the notions of computation tree/traversal Ong was able to show (LICS06) that this result still holds in the unrestricted case.
- ▶ Studying the effect of syntactic restrictions on the game semantics model. e.g. One can show that **pointers are uniquely recoverable** in the game denotation of terms satisfying the safety restriction.

### Related works:

- ▶ Stirling recently proved decidability of higher-order pattern matching with a game-semantic approach relying on equivalent notions of computation tree and traversal.

# Outline

## The correspondence

- Game semantics

- Computation tree

- The Correspondence Theorem

- Example

- Composition

- Demo

## Applications

- Higher-order grammars

- Other applications






## Conclusion & Future Works

# Conclusion & Future Works

- ▶ **Conclusion:** a new **concrete** way to present game semantics based on the theory of **traversals**.
- ▶ **Future works:**
  - ▶ Extend the correspondence to PCF and Idealized Algol;
  - ▶ Can we generalize CPDA to give an automata characterization of the simply-typed lambda calculus?
  - ▶ Consider the Reachability problem in the traversal setting,
  - ▶ Complexity: characterization of space-complexity classes by analyzing the length of the traversals? (See Kazushige Terui's work.);
  - ▶ Implement other transformations: from CPDA to HORS and from safe HORS to PDA;
  - ▶ Implement the MSO decision procedure.



## Bibliography

-  S. Abramsky and G. McCusker  
Game semantics, Lecture notes.  
*In Proceedings of the 1997 Marktoberdorf Summer School. 1998.*
-  W. Blum and C.-H. L. Ong  
Local computation of beta-reduction  
Technical report. University of Oxford, 2008.
-  M. Hague, A.S. Murawski, C.-H. L. Ong and O. Serre  
Collapsible pushdown automata and recursive schemes.  
To appear, LICS2008.
-  C.-H. Luke Ong  
On model-checking trees generated by higher-order recursion schemes.  
*In Proceedings of LICS2006.*
-  C. Stirling  
A game-theoretic approach to deciding higher-order matching.  
*In Proceedings of ICALP2006.*