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

BCTCS, 8 April 2008

Overview

- Game-semantic models are abstract *i.e.* independent of the syntax of the denotated 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.

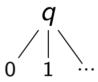
Outline

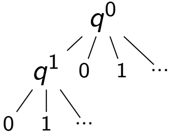
Outline

Game semantics

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

- 2 players: Opponent (system) and Proponent (program)
- The term type induces an arena defining the possible moves

  $[\![\mathbb{N}]\!] =$ 

  $[\![\mathbb{N} \rightarrow \mathbb{N}]\!] =$ 

- 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 $[\![M]\!]$, is a strategy for P.
- $[\![7 : \mathbb{N}]\!] = \{\epsilon, q, q\ 7\}$
  $[\![\mathrm{succ} : \mathbb{N} \rightarrow \mathbb{N}]\!] = \mathit{Pref}(\{q^0 q^1 n(n+1) \mid n \in \mathbb{N}\})$
- Compositionality: $[\![\mathrm{succ}\ 7]\!] = [\![\mathrm{succ}]\!]; [\![7]\!]$

Game semantics: composition

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

$$\text{“ } \sigma ; \mu = (\sigma \| \mu) \upharpoonright A, C \text{ ”}$$

- 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

$$\text{“ } \sigma; \mu = (\sigma \| \mu) \upharpoonright A, C \text{ ”}$$

- 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.

Outline

Computation tree

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

Computation tree of M is the AST of its η-long normal form.

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

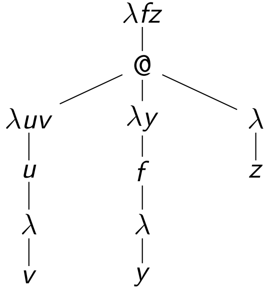
Example:

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

Its η-long normal form is

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

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 $\circledast$;
  - a lambda node is enabled by its parent node;
  - an @-node has no enabler.
- Distinction between external nodes $N^{\circledast\vdash}$(hereditarily justified by the root) and the internal nodes $N^{@\vdash}$ (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:
  - λ-nodes $\equiv$ O-moves
  - @-nodes and variable-nodes $\equiv$ P-moves
- Notions of alternation, P-view, O-view, P-visibility and O-visibility.

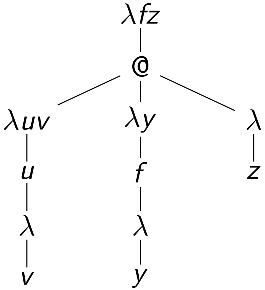
Traversals rules

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

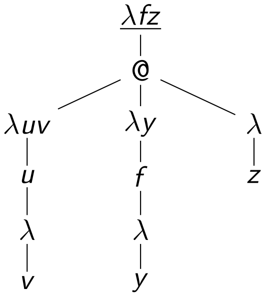
- (Empty) $\epsilon \in \mathcal{T}rav(M)$
- (Root) $\circledast \in \mathcal{T}rav(M)$
- (Lam) $t \cdot \lambda\overline{\xi} \in \mathcal{T}rav(M) \implies t \cdot \lambda\overline{\xi} \cdot n \in \mathcal{T}rav(M)$ where n is $\lambda\overline{\xi}$'s child and is justified by the only occurrence of its enabler in the P-view
- (App) $t \cdot @ \in \mathcal{T}rav(M) \implies t \cdot @ \cdot n \in \mathcal{T}rav(M)$ (n justified by $@$ with label 0).
- (ExtVar) $t \cdot x \in \mathcal{T}rav(M)$, $x \in N^{\circledast\vdash}_{\mathsf{var}} \implies t \cdot x \cdot n \in \mathcal{T}rav(M)$ for any λ-node n justified by some occurrence of its parent node in the O-view of t.
- (IntVar) $t \cdot n \cdot \lambda\overline{x} \ldots x_i \in \mathcal{T}rav(M)$ (x_i justified by $\lambda\overline{x}$ with label i), $x_i \in N^{@\vdash}_{\mathsf{var}} \implies$

  $t \cdot n \cdot \lambda\overline{x} \ldots x_i \cdot \lambda\overline{\eta_i} \in \mathcal{T}rav(M)$ (x_i justified by $\lambda\overline{x}$ with label i; $\lambda\overline{\eta_i}$ justified by n with label i).

Example of traversal

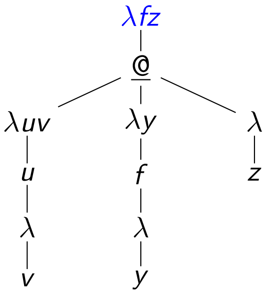


Example of traversal



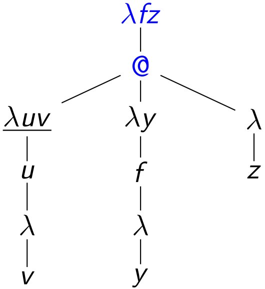
$t = \lambda fz$

Example of traversal



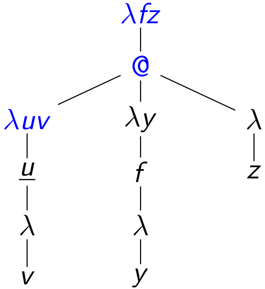
$t = \lambda fz \cdot @$

Example of traversal



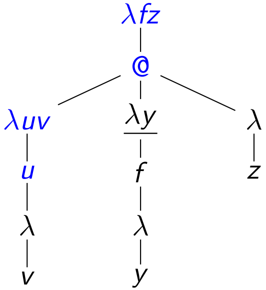
$$t = \lambda fz \cdot @ \cdot \lambda uv$$

Example of traversal



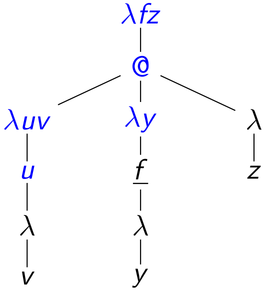
$$t = \lambda fz \cdot @ \cdot \lambda uv \cdot u$$

Example of traversal



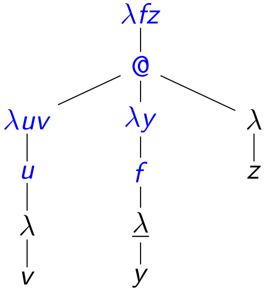
$$t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y$$

Example of traversal



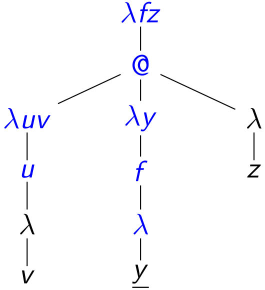
$$t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f$$

Example of traversal



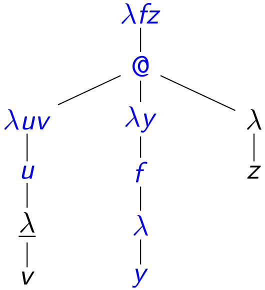
$$t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f \cdot \lambda$$

Example of traversal



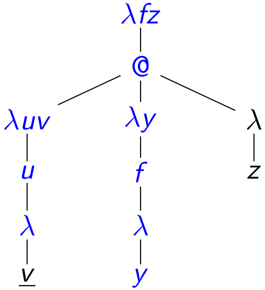
$$t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y$$

Example of traversal



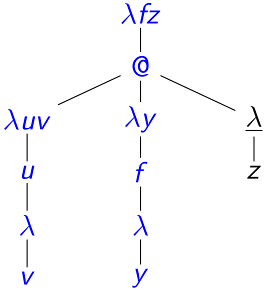
$$t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y \cdot \lambda$$

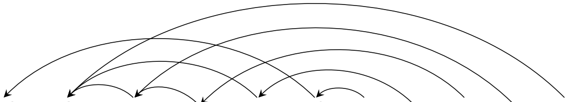
Example of traversal



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

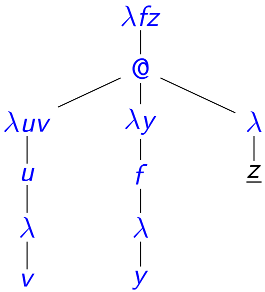
Example of traversal





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

Example of traversal



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

Operations on traversals

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

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

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

- *@-nodes removal:*

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

Operations on traversals

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

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

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

- *@-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 Correspondence Theorem

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

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

$$\varphi : \mathcal{T}rav(M)^{\upharpoonright \circledast} \xrightarrow{\cong} [\![M]\!] \ .$$

where

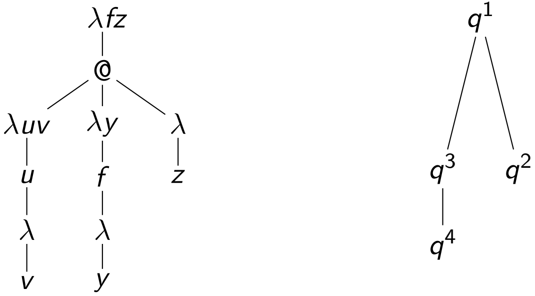
- $\mathcal{T}rav(M)$ = set of traversals of the computation tree of M
- $\mathcal{T}rav(M)^{\upharpoonright \circledast} = \{ t \upharpoonright t_0 \mid t \in \mathcal{T}rav(M) \}$
- $\mathcal{T}rav(M)^{-@} = \{ t - @ \mid t \in \mathcal{T}rav(M) \}$
- $[\![M]\!]$ = game-semantic denotation of M
- $\langle\!\langle M \rangle\!\rangle$ = revealed denotation of M.

More correspondences

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

Example: $\vdash \lambda f^{o \to o}.(\lambda u^{o \to o}.u)f : (o^4 \to o^3) \to o^2 \to o^1$

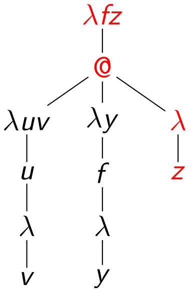
Left: computation tree. Right: arena.

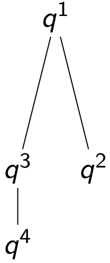


- $t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y \cdot \lambda \cdot v \cdot \lambda \cdot z$
- $\ulcorner t \urcorner = \lambda fz \cdot @ \cdot \lambda \cdot z$
- $\varphi(t \upharpoonright \lambda fz) = \varphi(\lambda fz \cdot f \cdot \lambda \cdot z) = q^1\ q^3\ q^4\ q^2 \in [\![M]\!]$.

Example: $\vdash \lambda f^{o\to o}.(\lambda u^{o\to o}.u)f : (o^4 \to o^3) \to o^2 \to o^1$

Left: computation tree. Right: arena.

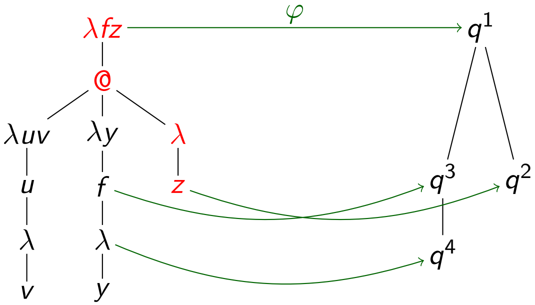




- $t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y \cdot \lambda \cdot v \cdot \lambda \cdot z$
- $\ulcorner t \urcorner = \lambda fz \cdot @ \cdot \lambda \cdot z$
- $\varphi(t \upharpoonright \lambda fz) = \varphi(\lambda fz \cdot f \cdot \lambda \cdot z) = q^1\ q^3\ q^4\ q^2 \in [\![M]\!]$.

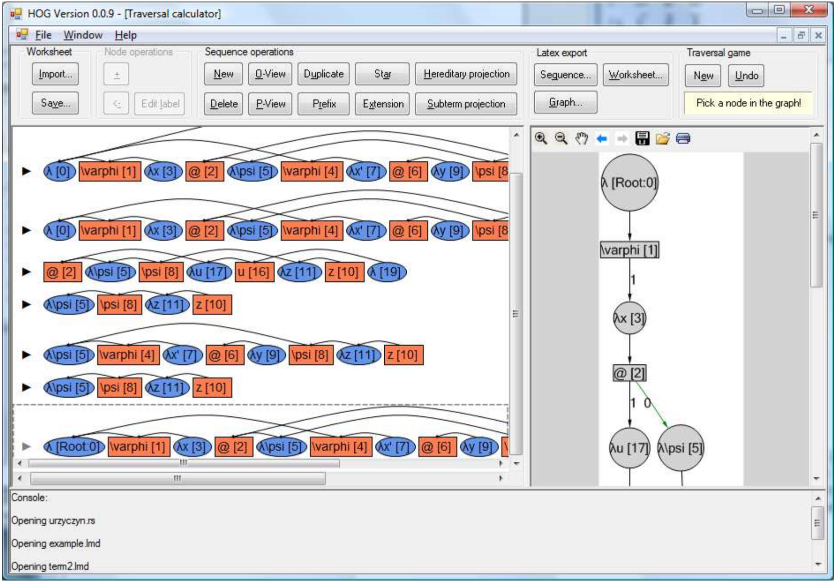
Example: $\vdash \lambda f^{o\to o}.(\lambda u^{o\to o}.u)f : (o^4 \to o^3) \to o^2 \to o^1$

Left: computation tree. Right: arena.



- $t = \lambda fz \cdot @ \cdot \lambda uv \cdot u \cdot \lambda y \cdot f \cdot \lambda \cdot y \cdot \lambda \cdot v \cdot \lambda \cdot z$
- $\ulcorner t \urcorner = \lambda fz \cdot @ \cdot \lambda \cdot z$
- $\varphi(t \upharpoonright \lambda fz) = \varphi(\lambda fz \cdot f \cdot \lambda \cdot z) = q^1\ q^3\ q^4\ q^2 \in [\![M]\!]$.

Tool demo


HOG Version 0.0.9 - [Traversal calculator]
File Window Help
Worksheet
Import...
Save...
Node operations
Edit label
Sequence operations
New
O-View
Duplicate
Star
Hereditary projection
Delete
P-View
Prefix
Extension
Subterm projection
Latex export
Sequence...
Worksheet...
Graph...
Traversal game
New
Undo
Pick a node in the graph!
λ [Root:0]
\varphi [1]
λx [3]
@ [2]
λu [17]
λ\psi [5]
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.* $\mathtt{new}\ \mathtt{x} := \lambda y.y\ \mathtt{in}\ x\ a$;
- 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 $\mathcal{O}(\mathrm{ord}\, M)$.

Outline

Applications, related works

- Studying infinite structures generated by higher-order programs.
- 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.

Applications, related works

- Studying infinite structures generated by higher-order programs.
- 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.

Applications, related works

- Studying infinite structures generated by higher-order programs.
- 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.

Applications, related works

- Studying infinite structures generated by higher-order programs.
- 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

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;
  - 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.);

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.*