Transition Systems over Games

Paul Blain Levy, University of Birmingham
Sam Staton, Radboud University Nijmegen

November 28, 2014

Outline

What is game semantics?

- A form of denotational semantics for many different language features.
- Game between P (Proponent, Patricia, the program)
- and O (Opponent, Oliver, the environment)

What is game semantics?

- A form of denotational semantics for many different language features.
- Game between P (Proponent, Patricia, the program)
- and O (Opponent, Oliver, the environment)
- Particularly good for modelling local state
  (Abramsky, Honda, McCusker, Reddy)

What is game semantics?

- A form of denotational semantics for many different language features.
- Game between P (Proponent, Patricia, the program)
- and O (Opponent, Oliver, the environment)
- Particularly good for modelling local state
  (Abramsky, Honda, McCusker, Reddy)
- because you don't see it in the semantics.

Example with integer state

Call-by-value λ-calculus, computation of type $\texttt{nat} \to \texttt{nat}$.

- P returns a function $\texttt{f} : \texttt{nat} \to \texttt{nat}$.
- O calls $\texttt{f}$ with argument 7.
- P returns 5.
- O calls $\texttt{f}$ with argument 7.
- P returns 29.

Example with higher-order state

Computation of type $(\mathtt{nat} \to \mathtt{nat}) \to \mathtt{nat}$.

- P returns $\mathtt{f}$.
- O calls $\mathtt{f}$ with argument $\mathrm{g} : \mathtt{nat} \to \mathtt{nat}$.
- P returns 7.
- O calls $\mathtt{f}$ with argument $\mathrm{g}' : \mathtt{nat} \to \mathtt{nat}$.
- P calls g with argument 2.
- O returns 3.
- P returns 5.

Example with higher-order state

Computation of type $(\mathtt{nat} \to \mathtt{nat}) \to \mathtt{nat}$.

- P returns $\mathtt{f}$.
- O calls $\mathtt{f}$ with argument $g : \mathtt{nat} \to \mathtt{nat}$.
- P returns 7.
- O calls $\mathtt{f}$ with argument $g' : \mathtt{nat} \to \mathtt{nat}$.
- P calls g with argument 2.
- O returns 3.
- P returns 5.

Arguments and return values that are functions
are represented as fresh names.

Example theorem

In λ-calculus with state and `callcc`:

- let M be a term without state
- let N be a term without `callcc`
- suppose $M \simeq N$.

Then there's a term P with neither state nor `callcc` such that

$$M \simeq P \simeq N$$

Example calculus (1)

Call-by-value λ-calculus with types

$$A ::= \quad 0 \mid A + A \mid 1 \mid A \times A \mid A \to A \mid \mathtt{X} \mid \mathtt{rec}\ \mathtt{X}.\, A$$

and general references.

Complications

- Two kinds of moves: calling a function (Question) and returning (Answer).
- Dealing with the call stack.

Example (2): calculus of no return

The target of the CPS transform.

$$A ::= \quad 0 \mid A + A \mid 1 \mid A \times A \mid \neg A \mid \mathtt{X} \mid \mathtt{rec}\ \mathtt{X}.\, A$$

Think: $\neg A = A \to 0$.

Terms are values $\Gamma \vdash^{\mathsf{v}} V : A$ and non-returning commands $\Gamma \vdash^{\mathsf{nc}} M$.

Operational semantics is the C-machine:
β-reduce until you reach $\mathtt{x}\, V$.

Ultimate pattern matching theorem

Every value has a unique decomposition into an ultimate pattern (tags) and a filling (functions).

Example $\langle \mathtt{inl}\ \mathtt{inr}\ \langle \lambda \mathrm{x}.\, M, \mathrm{y} \rangle, \mathtt{inl}\ \mathrm{y} \rangle$

Ultimate pattern is $\langle \mathtt{inl}\ \mathtt{inr}\ \langle -, - \rangle, \mathtt{inl}\ - \rangle$

Filling is $\lambda \mathrm{x}.\, M, \quad \mathrm{y}, \quad \mathrm{y}$.

Example (3): a untyped calculus

Make the recursive type $A \cong \neg(A \times A)$ into a calculus.

$$\begin{array}{rcl} V & ::= & \mathtt{x} \mid \lambda(\mathtt{x}, \mathtt{y}).\, M \\ M & ::= & V\,(V, V) \end{array}$$

Operational semantics is C-machine:
β-reduce until you reach $\mathtt{x}(V, V')$.

Our motivation

Denotational vs operational

- Traditional pointer-game semantics (Hyland, Ong et al, 1994 onwards) is compositional.
- Open (normal form) bisimulation (Sangiorgi et al, 1994 onwards) is operational but not obviously compositional.
- Very different but semantically the same.

Our motivation

Denotational vs operational

- Traditional pointer-game semantics (Hyland, Ong et al, 1994 onwards) is compositional.
- Open (normal form) bisimulation (Sangiorgi et al, 1994 onwards) is operational but not obviously compositional.
- Very different but semantically the same.

We wanted to combine these accounts

- Start with the intuitive operational description.
- Obtain compositionality as a theorem, not a definition.

Our motivation

Denotational vs operational

- Traditional pointer-game semantics (Hyland, Ong et al, 1994 onwards) is compositional.
- Open (normal form) bisimulation (Sangiorgi et al, 1994 onwards) is operational but not obviously compositional.
- Very different but semantically the same.

We wanted to combine these accounts

- Start with the intuitive operational description.
- Obtain compositionality as a theorem, not a definition.
- And treat fresh names in a rigorous yet unobtrusive way.

Our motivation

Denotational vs operational

- Traditional pointer-game semantics (Hyland, Ong et al, 1994 onwards) is compositional.
- Open (normal form) bisimulation (Sangiorgi et al, 1994 onwards) is operational but not obviously compositional.
- Very different but semantically the same.

We wanted to combine these accounts

- Start with the intuitive operational description.
- Obtain compositionality as a theorem, not a definition.
- And treat fresh names in a rigorous yet unobtrusive way.

We needed to think more carefully about transition systems.

Labelled transition system

Over a set

An LTS over a set L of actions consists of

- a set $\mathbb{S}$ of states
- a relation $\overset{a}{\rightsquigarrow}$ for each $a \in L$.

Labelled transition system

Over a set

An LTS over a set L of actions consists of

- a set $\mathbb{S}$ of states
- a relation $\overset{a}{\rightsquigarrow}$ for each $a \in L$.

What's odd

- Actions may represent outputs, inputs or sychronizations.
- The set of actions does not change over time.

Example: chess

Consider a system that plays chess.

Every state is in a position.

The position determines what actions are legitimate.

"The position is the type of the state."

Example: chess

Consider a system that plays chess.

Every state is in a position.

The position determines what actions are legitimate.

"The position is the type of the state."

Traditional LTS's are "untyped" or single-sorted.

Example: higher-order functions

Each player has an inventory of function-names they are allowed to call.

Position = two finite sets of function-names

Example: higher-order functions

Each player has an inventory of function-names they are allowed to call.

Position = two finite sets of function-names

In the semantics of a higher-order function call:

$$\texttt{f}(\lambda \mathrm{x}.\, M, \lambda \mathrm{x}.\, M')$$

P performs the move $\texttt{f}(-,-)$.
and O receives two fresh function-names b and b' for future use.

The new state contains the bindings

$$\begin{array}{rcl} b & \mapsto & \lambda \mathrm{x}.\, M \\ b' & \mapsto & \lambda \mathrm{x}.\, M' \end{array}$$

The Framework

	Single game $\mathcal{G}$	Tensor $\mathcal{G} \otimes \mathcal{G}'$	Transfer $\mathcal{G} \to \mathcal{H}$
Games			
Strategies			
Transition systems			
Relating strategies to transition systems		Compositionality theorems	

Game = bipartite graph

Definition

A game consists of

- a set of passive positions (O to move)
- a set of active positions (P to move)
- from each passive position P, a set of O-moves m, each with an active target position $P.m$
- from each active position Q, a set of P-moves n, each with a passive target position $Q.n$.

Notation

$P \circ\!\xrightarrow{m} Q$ O-move

$Q \bullet\!\xrightarrow{n} P$ P-move

Plays

Let P be a passive position. (The starting position.)

A play from position P is a sequence of moves

$$P \circ\!\xrightarrow{m_0} \cdot \bullet\!\xrightarrow{n_0} \cdot \circ\!\xrightarrow{m_1} \cdot \bullet\!\xrightarrow{n_1} \cdot$$

Strategies

A strategy tells P how to respond to any O-move

either playing a move or diverging.

Strategies

A strategy tells P how to respond to any O-move

either playing a move or diverging.

Definition

A strategy σ from passive position P is a set of passive-ending plays such that

- $\varepsilon \in \sigma$
- (prefix-closure) $smn \in \sigma \Rightarrow s \in \sigma$
- (determinacy) $tn, tn' \in \sigma \Rightarrow n = n'$

Small-step system over a game

Definition

- In each passive position, a set of passive states.
- In each active position, a set of active states.
- For each passive state x and O-move m, an active state $x@m$.
- Each active state y either
  - performs a P-move $y \overset{n}{\leadsto} x$
  - or performs a silent transition $y \leadsto y'$ (same position).

Derived notation

$$y \overset{n}{\Longrightarrow} x \quad \text{when } y \leadsto^{*} \overset{n}{\leadsto} x$$

$$y \Uparrow \quad \text{when } y \leadsto^{\omega}$$

Big-step system over a game

Definition

- In each passive position, a set of passive states.
- In each active position, a set of active states.
- For each passive state x and O-move m, an active state $x@m$.
- Each active state y either
  - performs a P-move $y \stackrel{n}{\Longrightarrow}$
  - or diverges $y \Uparrow$.

Passive system over a game

Definition

- In each passive position, a set of passive states.
- For each passive state x and O-move m,
  x responds to m by either
  - performing a P-move $x@m \xRightarrow{n}$
  - or diverging $x@m \Uparrow$.

From states to strategies

For a state x in passive position P, suppose

$$x = x_0 \quad x_0 @ m_0 \overset{n_0}{\rightsquigarrow} x_1 \quad x_1 @ m_1 \overset{n_1}{\rightsquigarrow} x_2 \quad \cdots$$

then the play $m_0, n_0, m_1, n_1 \cdots$ is a *trace* of x.

From states to strategies

For a state x in passive position P, suppose

$$x = x_0 \quad x_0 @ m_0 \overset{n_0}{\leadsto} x_1 \quad x_1 @ m_1 \overset{n_1}{\leadsto} x_2 \quad \cdots$$

then the play $m_0, n_0, m_1, n_1 \cdots$ is a *trace* of x.

The set of all traces of x is a strategy $[\![x]\!]$.

From states to strategies

For a state x in passive position P, suppose

$$x = x_0 \quad x_0 @ m_0 \overset{n_0}{\leadsto} x_1 \quad x_1 @ m_1 \overset{n_1}{\leadsto} x_2 \quad \cdots$$

then the play $m_0, n_0, m_1, n_1 \cdots$ is a *trace* of x.

The set of all traces of x is a strategy $[\![x]\!]$.

Big-step (or passive) bisimilarity implies trace equivalence
and conversely by determinism.

Categorical games

So far we've studied discrete games, having

- a set of passive positions
- a set of active positions.

Our examples are categorical games, having

- a category of passive positions
- a category of active positions.

Categorical games

So far we've studied discrete games, having

- a set of passive positions
- a set of active positions.

Our examples are categorical games, having

- a category of passive positions
- a category of active positions.

Example

- A position is a pair of finite sets of names $\Gamma \parallel \Delta$.
- A morphism $\Gamma \parallel \Delta \to \Gamma' \parallel \Delta'$
  is a pair of renamings $\Gamma \to \Gamma'$ and $\Delta' \to \Delta$.

Categorical games

So far we've studied discrete games, having

- a set of passive positions
- a set of active positions.

Our examples are categorical games, having

- a category of passive positions
- a category of active positions.

Example

- A position is a pair of finite sets of names $\Gamma \parallel \Delta$.
- A morphism $\Gamma \parallel \Delta \to \Gamma' \parallel \Delta'$
  is a pair of renamings $\Gamma \to \Gamma'$ and $\Delta' \to \Delta$.

Compositionality of renaming is naturality.

Tensor game $\mathcal{G} \otimes \mathcal{G}'$ (Lamarche)

We wish to play the two games concurrently.

- A passive position of $\mathcal{G} \otimes \mathcal{G}'$ is a pair of passive positions (P, P').
- O can choose which game to play in
- and P has to respond in the same game.
- So an active position has one active and one passive component.

Tensor strategies

- Given strategy σ from position P
- and σ' from position P'
- define $\sigma \otimes \sigma'$ from (P, P').
- It consists of all plays whose left projection is in σ and whose right projection is in σ'.

Tensor of transition systems

Given transition systems over $\mathcal{G}$ and $\mathcal{G}'$,

the tensor system has states (x, x').

Compositionality theorem for tensors

$$[\![(x, x')]\!] = [\![x]\!] \otimes [\![x']\!]$$

Transfer from $\mathcal{G}$ to $\mathcal{H}$

I am going to play the external game $\mathcal{H}$ (chess) against external-O.

In my attic lives a player of the internal game $\mathcal{G}$ (draughts) called internal-P.

I shall transfer moves between the two games using a transfer.

Transfer from $\mathcal{G}$ to $\mathcal{H}$

I am going to play the external game $\mathcal{H}$ (chess) against external-O.

In my attic lives a player of the internal game $\mathcal{G}$ (draughts) called internal-P.

I shall transfer moves between the two games using a transfer.

Secret

A transfer is a total passive system over

$$\mathcal{G} \multimap \mathcal{H} = (\mathcal{G} \otimes \mathcal{H}^{\perp})^{\perp}$$

Positions and linkers

At any time, either

- both games are in passive position and I'm waiting for external-O
- or both games are in active position and I'm waiting for internal-P.

Positions and linkers

At any time, either

- both games are in passive position and I'm waiting for external-O
- or both games are in active position and I'm waiting for internal-P.

My own state is called a linker

$$\begin{array}{cl} P & \text{internal position} \\ \downarrow & \\ R & \text{external position} \end{array}$$

Example: a binary transfer

λ**Game** is the game for λ-calculus.

- A position is two finite sets of function-names.

Example: a binary transfer

λ**Game** is the game for λ-calculus.

- A position is two finite sets of function-names.

We define a transfer from $\lambda\mathbf{Game} \otimes \lambda\mathbf{Game} \to \lambda\mathbf{Game}$.

Intended purpose of the transfer

To provide a semantic counterpart to syntactic substitution.

Example: a binary transfer

λ**Game** is the game for λ-calculus.

- A position is two finite sets of function-names.

We define a transfer from $\lambda\mathbf{Game} \otimes \lambda\mathbf{Game} \to \lambda\mathbf{Game}$.

Intended purpose of the transfer

To provide a semantic counterpart to syntactic substitution.

What's a linker in this transfer?

A function saying that certain names in one game correspond to certain names in the other.

Responding to an external O-move

Given a passive linker
and external O-move

$$\begin{tikzcd} P \arrow[d, "h"'] & \\ R \arrow[r, "m'"] & {} \end{tikzcd}$$

I play either

$$\begin{tikzcd} P \arrow[r, dotted, "m"'] \arrow[d, "h"'] & \cdot \arrow[d, dotted, "k"] \\ R \arrow[r, "m'"] & \cdot \end{tikzcd}$$

O-move
square

or

$$\begin{tikzcd} P \arrow[rr, equal] \arrow[d, "h"'] & & P \arrow[d, dotted, "h'"] \\ R \arrow[r, "m'"] & \cdot\bullet \arrow[r, dotted, "n'"] & \cdot \end{tikzcd}$$

external
square

Responding to an internal P-move

Given an active linker
and internal P-move

$$\begin{tikzcd} Q \arrow[r, "n"'] \arrow[d, "k"'] & {} \\ S & \end{tikzcd}$$

I play either

$$\begin{tikzcd} Q \arrow[r, "n"'] \arrow[d, "k"'] & \cdot \circ \arrow[r, dotted, "m"'] & \cdot \arrow[d, dotted, "k'"] \\ S \arrow[rr, equal] & & S \end{tikzcd}$$

internal
square

or

$$\begin{tikzcd} Q \arrow[r, "n"'] \arrow[d, "k"'] & Q.n \arrow[d, dotted, "h"] \\ S \arrow[r, dotted, "n'"] & S.n' \end{tikzcd}$$

P-move
square

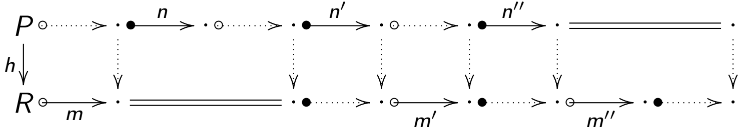
What is a transfer?

A transfer from $\mathcal{G}$ to $\mathcal{H}$ consists of

- a collection of passive linkers
- a collection of active linkers
- a collection of interaction squares (of four kinds) saying how to respond to every external O-move and every internal P-move.

An interaction sequence from a linker is a sequence of interaction squares.

internal play



external play

Transferring strategies

Given a transfer $\mathcal{O}$ from $\mathcal{G}$ to $\mathcal{H}$

and a linker $h : P \rightarrow R$,

each strategy σ from P gives a strategy $\mathcal{O}(\sigma)$ from R

viz. the set of all external plays of interaction sequences from h whose internal play is in σ.

Compositionality theorem for transfers

Given small-step systems over $\mathcal{G}$ and $\mathcal{H}$

and a transfer $\mathcal{O}$ from $\mathcal{G}$ to $\mathcal{H}$

and a linker-indexed relation $\mathcal{R}$, i.e.
for each linker $h : P \rightarrow R$
a relation $\mathcal{R}_h$ from states in position P to states in position R.

Suppose $\mathcal{R}$ is a stepped bisimulation across $\mathcal{O}$.

If $x \, \mathcal{R}_h \, y$ then $[\![y]\!] = \mathcal{O}([\![x]\!])$.

What's a stepped bisimulation?

For a linker-indexed relation to be a stepped bisimulation,

it must be preserved across each kind of square and across silent transitions.

What's a stepped bisimulation?

For a linker-indexed relation to be a stepped bisimulation,

it must be preserved across each kind of square and across silent transitions.

But that's not enough.

The danger

$x\,\mathcal{R}_k\,y$ and we have infinitely many internal squares

$$\begin{tikzcd}
\cdot \arrow[r] \arrow[d, "k"'] & \cdot \arrow[r, dotted] & \cdot \arrow[r] \arrow[d] & \cdot \arrow[r, dotted] & \cdot \arrow[d] & \cdots \\
\cdot \arrow[rr, equal] & & \cdot \arrow[rr, equal] & & \cdot &
\end{tikzcd}$$

Then the transfer predicts that y diverges, but it might not.

To rule this out, predicates $(U_k^i)_{i\in\mathbb{N}}$ bound the number of internal squares.

What have we gained?

To describe a game semantics we first give a transition system over a game.

Then for each term constructor we give a transfer, and a stepped bisimulation to demonstrate its correctness.

We only need to talk about individual moves. Plays and strategies are handled by our compositionality theorems.

Further directions

- Game semantics for many different languages
- $!\mathcal{G}$ for multiple threads (Hyland)
- $*$-autonomous bicategory of games and transfers
- Equations between operations on strategies arising from transfers
- Nondeterminism, probability, ...
- Concurrency?