Transition Systems over Games

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

April 13, 2014

Denotational game semantics

- Game semantics with justification pointers
- Defined compositionally
- The “traditional” way of presenting game semantics
- Successful for many language features, especially state
- Hyland and Ong; Abramsky, Honda and McCusker; . . .

Operational game semantics

- Sangiorgi 1994; many people in 2007
- Semantics given by a labelled transition system
- Nodes are configurations built from terms
- Transitions use operational semantics of open terms
- Highly intuitive: it's obvious that a term's meaning is what you expect it to be
- Not obviously compositional.

Combining Operational and Denotational

- Want to show that the semantics given by the transition system is actually compositional.
- That provides denotational semantics.
- E.g. Rathke and Jeffrey; Laird; Ghica and Tzevelekos
- I tried to do this and found it hard, at least in a way that brought out certain aspects of the story.

Some things we wanted

- Transition systems should be coalgebras for a suitable functor.
- Treatment of names should be rigorous yet unobtrusive.
- Renaming should be done in a presheaf category.
- The denotational model should be a category enriched in Lamarche games.
- Adaptation to nondeterministic versions should be straightforward,
- whether we use bisimilarity, finite traces, infinite traces ...

Not seeking to include concurrent games.

What we found

- Current accounts of (denotational/operational/combined) game semantics are actually two stories mixed together.
- The first story is a general framework: a collection of categorical definitions and theorems regarding games, transition systems, transfers and bisimulations
- The second story is describing the particular games, transition systems, etc.
- The second story is different for each language/model we study.
- The second story is only about individual moves.
- Plays and strategies are treated in the first story.

The Framework

	Categorical game $\mathcal{G}$	Tensor $\mathcal{G} \otimes \mathcal{H}$	Multiple threads $\mathcal{G} \to \mathcal{H}$	Transfer $!\mathcal{G}$
Games				
Strategies				
Transition systems				
Relating strategies to transition systems				Small-step bisimulation across a transfer

Example calculus

The call-by-value recursive type $A = (A \times A) \to 0$ gives an untyped (or uni-typed) calculus, with syntax

Value	V	$::=$	$\mathtt{x} \mid \lambda(\mathtt{x}, \mathtt{y}).M$
Nonreturning command	M	$::=$	$V(V, V)$

with judgements $\Gamma \vdash^{\mathsf{v}} V$ and $\Gamma \vdash^{\mathsf{nc}} M$.

Operational semantics is β-reduction.

Example P-move

Consider the command

$$\mathtt{x}, \mathtt{y} \vdash^{\mathsf{nc}} (\lambda(\mathtt{s}, \mathtt{t}).\mathtt{s}(\mathtt{t}, \mathtt{y}))(\mathtt{x}, \lambda(\mathtt{p}, \mathtt{q}).(\lambda(\mathtt{u}, \mathtt{v}).\mathtt{u}(\mathtt{v}, \mathtt{q}))(\mathtt{p}, \mathtt{x}))$$

P performs β-reduction, reaching $\mathtt{x}(\lambda(\mathtt{p}, \mathtt{q}).(\lambda(\mathtt{u}, \mathtt{v}).\mathtt{u}(\mathtt{v}, \mathtt{q}))(\mathtt{p}, \mathtt{x}), \mathtt{y})$ So she calls $\mathtt{x}$, passing two functions that we'll call b_0 and b_1.

Actually, $\begin{array}{lll} b_0 & \text{is} & \lambda(\mathtt{p}, \mathtt{q}).(\lambda(\mathtt{u}, \mathtt{v}).\mathtt{u}(\mathtt{v}, \mathtt{q}))(\mathtt{p}, \mathtt{x}) \\ b_1 & \text{is} & \mathtt{y} \end{array}$

Example O-move

Suppose now O calls b_0,

passing two functions that we'll call $\mathtt{w}_0$ and $\mathtt{w}_1$.

Now P executes $(\lambda(\mathtt{p},\mathtt{q}).(\lambda(\mathtt{u},\mathtt{v}).\mathtt{u}(\mathtt{v},\mathtt{q}))(\mathtt{p},\mathtt{x}))(\mathtt{w}_0,\mathtt{w}_1)$

Summary of play

Each player has an inventory of function-names, that grows over time.

A player moves by calling one of them. Then the other player acquires two fresh names.

We keep track of what each O-name is in reality

so that an O-moves gives rise to a command for P to execute.

Digression: fresh names

To present our transition system, should we

- allow arbitrary fresh names to be generated?
- or use a fixed schedule?

The latter turns out to make everything simpler.

Gen-set: a set of names with generation

Definition

A gen-set consists of

- a set A
- a set of permitted finite subsets of A
- for any permitted subset R, an element $\nu R \in A$ such that
  - $\nu R \notin R$
  - $R^{+} \overset{\mathrm{def}}{=} R \cup \{\nu R\}$ is permitted.

Examples of gen-sets

1. $\mathbb{N}$, with $\{0, \ldots, n-1\}$ permitted. Cf. our O-names $b_0, b_1, b_2, \ldots$.
2. $B + \mathbb{N}$, with $U + \{0, \ldots, n-1\}$ permitted. Cf. our P-names $\mathtt{x}, \mathtt{y}, \mathtt{w}_0, \mathtt{w}_2, \mathtt{w}_3, \ldots$.

   The universe of sets is a gen-class with all finite subsets permitted.

Fix a gen-set of P-names and one of O-names.

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

$$p \circ\!\xrightarrow{m} q \qquad \text{for O-moves}$$

$$q \bullet\!\xrightarrow{n} p \qquad \text{for P-moves}$$

Example game λ**Game**

- A passive position is $\Gamma \parallel \Delta$, permitted sets of P-names and O-names respectively
- An active position is $\Gamma \parallel \Delta$, permitted sets of P-names and O-names respectively
- An O-move from $\Gamma \parallel \Delta$ is $b \in \Delta$, with result position $\Gamma^{+2} \parallel \Delta$
- A P-move from $\Gamma \parallel \Delta$ is $\mathtt{x} \in \Gamma$, with result position $\Gamma \parallel \Delta^{+2}$.

Plays and Strategies

A play from position p is a sequence of moves.

A (deterministic) strategy from p is a set of passive-ending plays subject to prefix-closure and determinacy.

Transition systems

Definition

A small-step transition system $\mathbb{S}$ over $\mathcal{G}$ consists of

- a set of nodes in each position, written $\mathbb{S}^{\mathrm{pass}}\, p$ or $\mathbb{S}^{\mathrm{act}}\, q$
- functions

$$\begin{aligned} \mathbb{S}^{\mathrm{pass}}\, p &\to \prod_{m \in \mathrm{Omove}\, p} \mathbb{S}^{\mathrm{act}}\, p.m \\ \mathbb{S}^{\mathrm{act}}\, q &\to \mathbb{S}^{\mathrm{act}}\, q + \sum_{n \in \mathrm{Pmove}\, q} \mathbb{S}^{\mathrm{pass}}\, q.n \end{aligned}$$

Notation

$x@m$	O-move
$y \rightsquigarrow z$	silent transition
$y \overset{n}{\rightsquigarrow} x$	P-move
$y \overset{n}{\Longrightarrow} x$	when $y \rightsquigarrow^{*} \overset{n}{\rightsquigarrow} x$

Example: transition system λ**Syst** over λ**Game**

- Passive node in position $\Gamma \parallel \Delta$ is a collection of values in context Γ

$$(V_a)_{a \in \Delta}$$

- Active node in position $\Gamma \parallel \Delta$ is a collection of values and a command in context Γ

$$(V_a)_{a \in \Delta} \blacktriangleright M$$

- O-move a applies a to fresh names
- Silent transition if the command β-reduces
- P-move $\mathtt{x}$ if the command is $\mathtt{x}(V, V')$.

From nodes to strategies

Each node x in position p

has an operational meaning $[x]$, a strategy from p.

This is the set of plays possible from p.

The category of positions

Passive positions form a category:

a morphism from $\Gamma \parallel \Delta$ to $\Gamma' \parallel \Delta'$ consists of functions $\Gamma \to \Gamma'$ and $\Delta' \to \Delta$

This gives a notion of categorical game $\mathcal{G}$.

Everything is functorial and natural.

Tensor of games $\mathcal{G}$ and $\mathcal{H}$

A passive position of $\mathcal{G} \otimes \mathcal{H}$ is a pair of passive positions (p, p').

O can choose to move on left or on right.

So an active position has one active side and one passive side.

We can also form the tensor of strategies and of transition systems.

Tensor compositionality:

$$[(x, y)] = [x] \otimes [y]$$

Example operation on nodes

Given an active node r in position $\mathtt{x}, \mathtt{y} \parallel$

and passive node s in position $\mathtt{z} \parallel a$

we obtain an active node $F(r, s)$ in position $\mathtt{x}, \mathtt{z} \parallel$

by substitution.

Operational researchers just want to show that F preserves bisimilarity (or trace equivalence), but to connect to denotational semantics we need more.

Wanted: a corresponding operation on nodes

First task

given an active left strategy σ from $\mathtt{x}, \mathtt{y} \parallel$

and passive right strategy τ from $\mathtt{z} \parallel a$

define an active strategy $G(\sigma, \tau)$ from $\mathtt{x}, \mathtt{z} \parallel$.

Second task

Show $[F(x, y)] = G([x], [y])$

Compositionality is a theorem, not a definition

Wanted: a corresponding operation on nodes

First task

given an active left strategy σ from $\mathtt{x}, \mathtt{y} \parallel$

and passive right strategy τ from $\mathtt{z} \parallel a$

define an active strategy $G(\sigma, \tau)$ from $\mathtt{x}, \mathtt{z} \parallel$.

Solution: TRANSFER

Second task

Show $[F(x, y)] = G([x], [y])$

Solution: BISIMULATION ACROSS A TRANSFER

Transferring moves

Three interfaces: left, right and external.

Awaiting external O

All three interfaces are passive.

When external O moves, generating two P-names,
this is copied to an external P-move, generating two O-names
or a left O-move or a right O-move, generating two P-names.

Awaiting left P (or right P)

Left and external interfaces are active; right is passive.

When left P moves, generating two O-names
this is copied to a right O-move, generating two P-names
or to an external P-move, generating two O-names.

Each name we generate is copied from a name we see generated.

The linker

For left position $\Gamma_l \parallel \Delta_l$
and right position $\Gamma_r \parallel \Delta_r$
and external position $\Gamma_e \parallel \Delta_e$,

a linker is a function

$$\Gamma_{\mathrm{l}} + \Gamma_{\mathrm{r}} + \Delta_{\mathrm{e}} \to \Delta_{\mathrm{l}} + \Delta_{\mathrm{r}} + \Gamma_{\mathrm{e}}$$

mapping each name to one at a different interface.

This says where each name originated from. It is essential to record this information in order to transfer moves properly.

Transfers

A transfer $\mathcal{G} \to \mathcal{H}$ is given by

- a collection of linkers from $\mathcal{G}$ positions to $\mathcal{H}$ positions (bimodule)
- a program that, given a linker and a move, produces another move and a new linker.

Example: our transfer $\lambda\mathbf{Comp} : \lambda\mathbf{Game} \otimes \lambda\mathbf{Game} \to \lambda\mathbf{Game}$.

The program can be represented as a collection of squares, such as

$$\begin{tikzcd}
(p_{\mathrm{l}}, p_{\mathrm{r}}) \arrow[r, dotted, "\mathsf{inl}\ b"] \arrow[d, "h"'] & ((\Gamma_{\mathrm{l}}^{+2} \parallel \Delta_{\mathrm{l}}), p_{\mathrm{r}}) \arrow[d, dotted, "{h[\mathsf{inl}\ \nu_i\Gamma_{\mathrm{l}} \mapsto \mathsf{ine}\,\nu_i\Gamma_{\mathrm{e}}]_{i=0,1}}"] \\
p_{\mathrm{e}} \arrow[r, "a"'] & \Gamma_{\mathrm{e}}^{+2} \parallel \Delta_{\mathrm{e}}
\end{tikzcd}$$

where $h(\mathsf{ine}\,a) = \mathsf{inl}\ \ b$

Interaction sequences

We can put squares together to make interaction sequences:

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

The internal (upper) edge is a play in $\mathcal{G}$, and the external (lower) edge is a play in $\mathcal{H}$.

This gives an operation on strategies.

The external node

As our process of transfer develops, there is always a node at the left and right interfaces

and an external node constructed from them by chains of substitution.

When we copy moves between left and right, the external node is unchanged.

So we have to rule out an infinite consecutive sequence of such steps, to guarantee progress.

Bisimulation across a transfer

We introduce a new kind of bisimulation across a transfer.

It is small-step with a progress condition.

Example: our construction of the external node.

Transfer compositionality theorem

- Given a bisimulation $\mathcal{R}$ across a transfer t.
- For linker h
- if $\mathcal{R}_h$ relates node x to node y
- then $[y]$ is obtained from $[x]$ by the strategy operation at h.