Compositional Game Theory in Type Theory

Fredrik Nordvall Forsberg
University of Strathclyde, Glasgow

TYPES 2019, Oslo, 14 June

What is economic game theory?

What is economic game theory?

The theory of interacting "rational" agents.

Players make observations and then make choices.

Choices of all players determine payoffs.

What is economic game theory?

The theory of interacting "rational" agents.

Players make observations and then make choices.

Choices of all players determine payoffs.

Players want to maximise their payoff.

Fundamental concept: equilibrium strategies.

What is economic game theory?

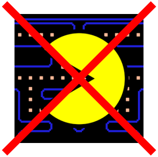
The theory of interacting “rational” agents.

Players make observations and then make choices.

Choices of all players determine payoffs.

Players want to maximise their payoff.

Fundamental concept: equilibrium strategies.



Example: penalty shootout

Choices $\Sigma = \{L, R\}^2$.

Payoffs $u : \Sigma \to \mathbb{R}^2$ with $u(a,b) = \begin{cases} (\;\;1, -1) & \text{if } a \neq b \\ (-1, \;\;1) & \text{if } a = b \end{cases}$

Example: penalty shootout

Choices $\Sigma = \{L, R\}^2$.

Payoffs $u : \Sigma \to \mathbb{R}^2$ with $u(a, b) = \begin{cases} (\ \ 1, -1) & \text{if } a \neq b \\ (-1, \ \ 1) & \text{if } a = b \end{cases}$

No (deterministic) equilibria.

The problem of scaling

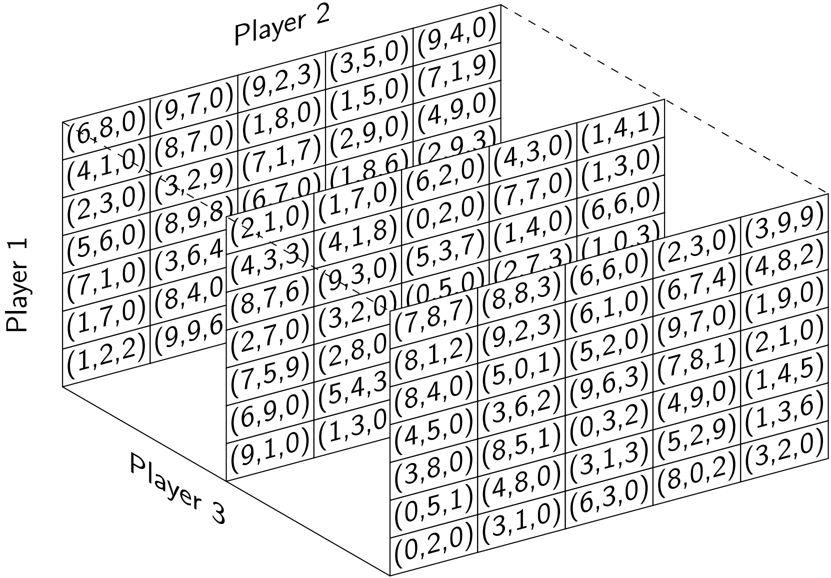
The problem of scaling

Player 2

Player 1

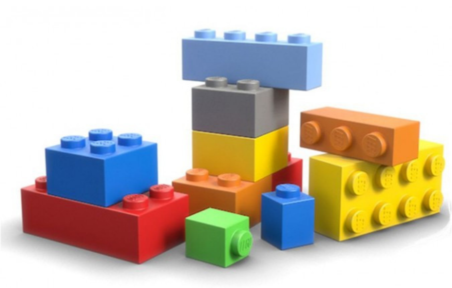
(5,2)	(1,0)	(4,0)	(3,6)	(8,1)	(0,9)	(2,5)	(0,4)	(6,3)	(8,0)
(6,2)	(3,1)	(2,5)	(8,3)	(5,8)	(6,0)	(3,8)	(9,6)	(6,5)	(8,2)
(8,5)	(9,7)	(3,6)	(8,1)	(4,7)	(2,0)	(0,6)	(2,9)	(0,4)	(5,2)
(6,2)	(3,1)	(4,0)	(7,7)	(2,7)	(0,7)	(7,1)	(9,5)	(3,8)	(6,7)
(1,8)	(9,2)	(5,9)	(2,1)	(2,2)	(8,2)	(8,6)	(1,4)	(0,2)	(0,7)
(5,9)	(8,4)	(5,8)	(1,8)	(2,7)	(0,2)	(7,1)	(2,6)	(6,3)	(0,0)
(8,8)	(0,1)	(9,1)	(3,5)	(5,8)	(6,7)	(2,9)	(6,9)	(8,2)	(3,4)
(7,9)	(6,9)	(5,7)	(4,7)	(7,0)	(3,8)	(5,8)	(9,2)	(7,1)	(8,3)
(3,9)	(6,4)	(7,7)	(5,4)	(1,7)	(9,0)	(4,8)	(4,9)	(6,4)	(0,5)
(1,2)	(1,2)	(3,5)	(6,3)	(9,3)	(2,9)	(5,2)	(8,7)	(0,3)	(5,1)
(9,1)	(0,1)	(8,8)	(2,4)	(4,6)	(1,0)	(6,0)	(2,6)	(5,7)	(3,9)

The problem of scaling



Building games compositionally

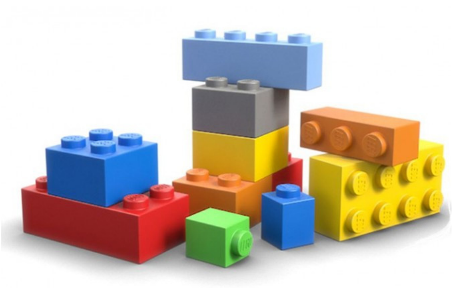
Goal: Instead of making sense of large games *post facto*, construct them from smaller, already understood games.



Building games compositionally

Goal: Instead of making sense of large games *post facto*, construct them from smaller, already understood games.

Trade-offs needed, because of emergent behaviour.

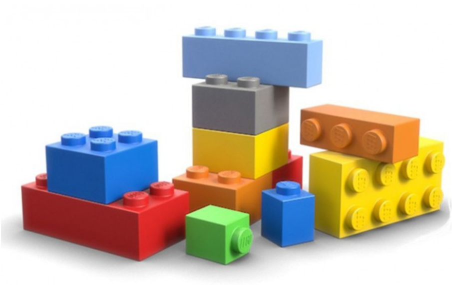


Building games compositionally

Goal: Instead of making sense of large games *post facto*, construct them from smaller, already understood games.

Trade-offs needed, because of emergent behaviour.

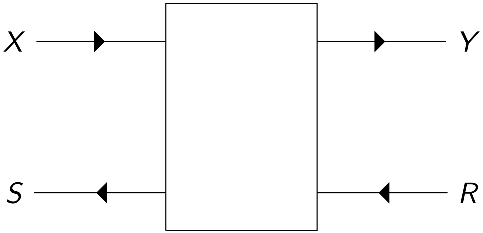
Methods: Category theory (for compositionality), type theory (for precision; this work).




The open games framework

Open games [Hedges 2016]

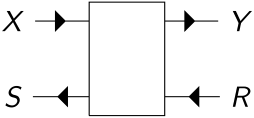
From the outside



$X \in \mathrm{Set}$	state of the game	$Y \in \mathrm{Set}$	moves of the game
$S \in \mathrm{Set}$	coutility type	$R \in \mathrm{Set}$	utility type

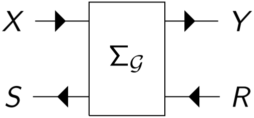
Open games

Inside the box



Open games

Inside the box

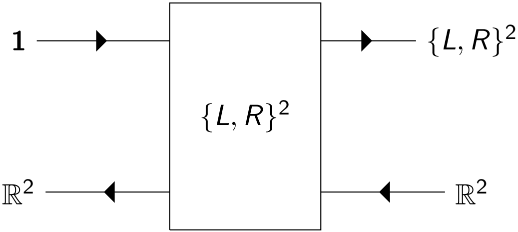


Definition

An open game $\mathcal{G} = (\Sigma_{\mathcal{G}}, P_{\mathcal{G}}, C_{\mathcal{G}}, E_{\mathcal{G}}) : (X, S) \to (Y, R)$ consists of:

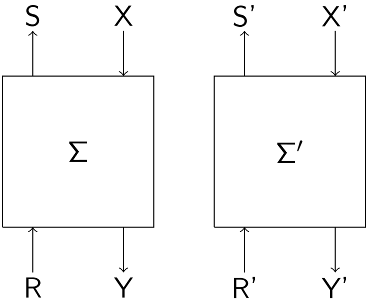
- a set $\Sigma_{\mathcal{G}}$ of strategy profiles,
- a play function $P_{\mathcal{G}} : X \to \Sigma_{\mathcal{G}} \to Y$,
- a coutility function $C_{\mathcal{G}} : X \to \Sigma_{\mathcal{G}} \to R \to S$, and
- an equilibrium function $E_{\mathcal{G}} : X \to (Y \to R) \to \mathscr{P}(\Sigma_{\mathcal{G}})$.

Example: penalty shootout as an open game

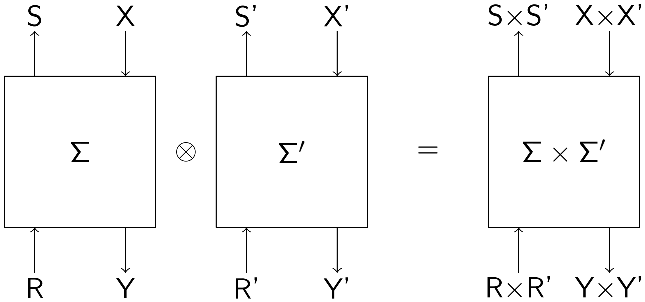


$$\begin{aligned}
P(x, \sigma) &= \sigma \\
C(x, \sigma, r) &= r \\
(a, b) \in E(x, k) \text{ iff } & \pi_1(k(a, b)) \geq \pi_1(k(\bar{a}, b)) \text{ and} \\
& \pi_2(k(a, b)) \geq \pi_2(k(a, \bar{b}))
\end{aligned}$$

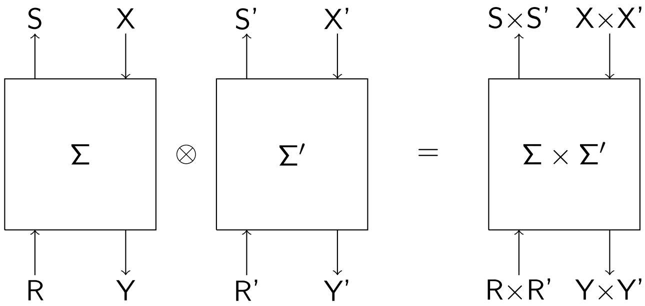
Parallel composition of open games



Parallel composition of open games



Parallel composition of open games



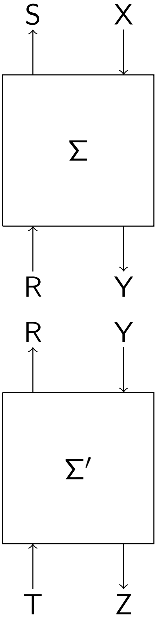
Proposition

The penalty shootout open game can be built as $P_1 \otimes P_2$, where

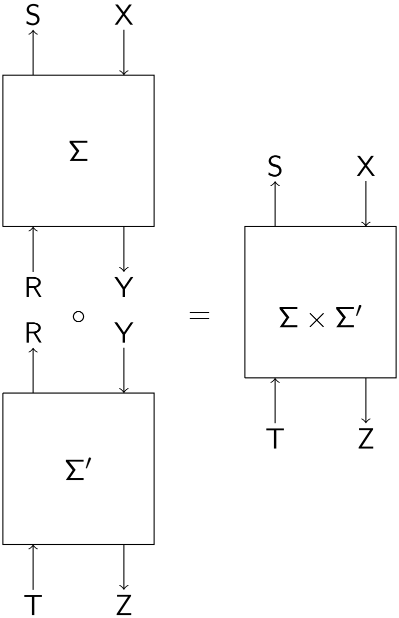
$$P_1, P_2 : (\mathbf{1}, \mathbb{R}) \to (\{L, R\}, \mathbb{R})$$

with $\Sigma_{P_i} = \{L, R\}$, and $a \in E_{P_i}(x, k)$ iff $a \in \arg\max_{x \in \Sigma}\{k(x)\}$.

Sequential composition



Sequential composition



Symmetric monoidal structure

Theorem ([Ghani, Hedges, Winschel, Zahn 2018])

(i) *The collection of pairs of sets, with open games* $\mathcal{G} : (X, S) \to (Y, R)$ *as morphisms, forms a symmetric monoidal category* Game.

Symmetric monoidal structure

Theorem ([Ghani, Hedges, Winschel, Zahn 2018])

(i) *The collection of pairs of sets, with open games $\mathcal{G} : (X, S) \to (Y, R)$ as morphisms, forms a symmetric monoidal category* Game*.*

(ii) *There is a identity-on-objects functor*

$$\iota : \mathsf{Set} \times \mathsf{Set}^{\mathsf{op}} \to \mathsf{Game}$$

with

$$P_{\iota(f,g)}(x, \sigma) = f(x) \qquad\qquad C_{\iota(f,g)}(x, \sigma, r) = g(r). \quad \square$$

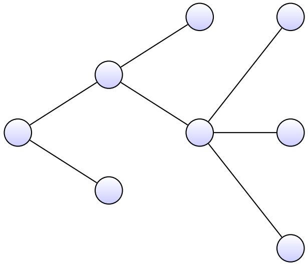
More structure?

Can we construct e.g. coproducts of games? (For a natural notion of morphisms between games.)

More structure?

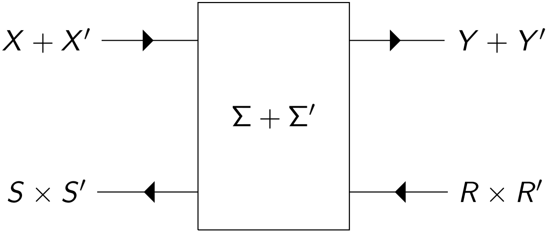
Can we construct e.g. coproducts of games? (For a natural notion of morphisms between games.)

Game-theoretic motivation: Games with *external* choice, e.g. later rounds depend on choices in previous rounds.



Coproduct construction attempts

First attempt:



$$P_{\mathcal{G}+\mathcal{G}'} : (X + X') \times (\Sigma + \Sigma') \to (Y + Y')$$

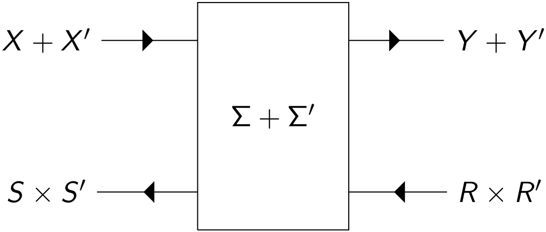
$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\, x)\,(\mathsf{inl}\,\sigma) = \{?_0 : Y + Y'\}$$

$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\, x)\,(\mathsf{inr}\,\sigma') = \{?_1 : Y + Y'\}$$

$$\vdots$$

Coproduct construction attempts

First attempt:



$$P_{\mathcal{G}+\mathcal{G}'} : (X + X') \times (\Sigma + \Sigma') \to (Y + Y')$$

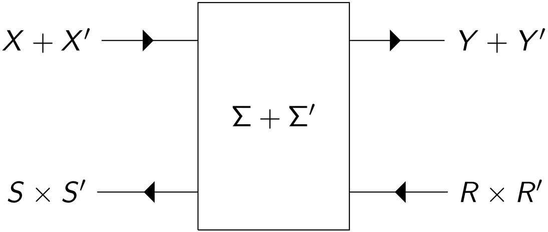
$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\, x)\,(\mathsf{inl}\, \sigma) = \mathsf{inl}\,(P_{\mathcal{G}}\, x\, \sigma)$$

$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\, x)\,(\mathsf{inr}\, \sigma') = \{?_1 : Y + Y'\}$$

$$\vdots$$

Coproduct construction attempts

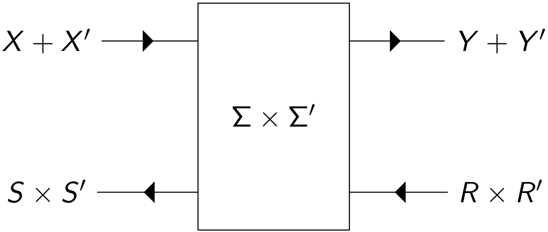
First attempt:



$$P_{\mathcal{G}+\mathcal{G}'} : (X + X') \times (\Sigma + \Sigma') \to (Y + Y')$$
$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\,x)\,(\mathsf{inl}\,\sigma) = \mathsf{inl}\,(P_{\mathcal{G}}\,x\,\sigma)$$
$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\,x)\,(\mathsf{inr}\,\sigma') = \;\text{???}\;\lightning$$
$$\vdots$$

Coproduct construction attempts

~~First~~ Second attempt:



$$P_{\mathcal{G}+\mathcal{G}'} : (X + X') \times (\Sigma \times \Sigma') \to (Y + Y')$$

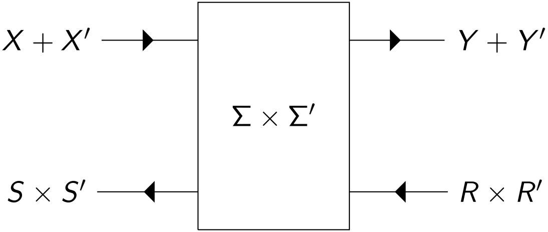
$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\, x)\,(\sigma, \sigma') = \{?_0 : Y + Y'\}$$

$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inr}\, x)\,(\sigma, \sigma') = \{?_1 : Y + Y'\}$$

$$\vdots$$

Coproduct construction attempts

~~First~~ Second attempt:

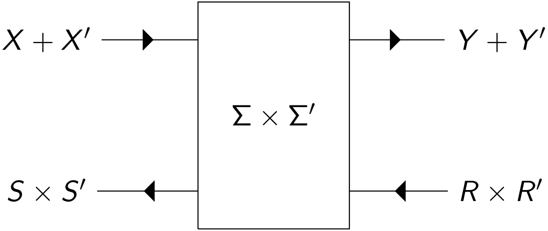


$$
\begin{aligned}
&P_{\mathcal{G}+\mathcal{G}'} : (X + X') \times (\Sigma \times \Sigma') \to (Y + Y') \\
&P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\, x)\,(\sigma, \sigma') = \mathsf{inl}\,(P_{\mathcal{G}}\, x\, \sigma) \\
&P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inr}\, x)\,(\sigma, \sigma') = \boxed{\{?_1 : Y + Y'\}}
\end{aligned}
$$

$$\vdots$$

Coproduct construction attempts

~~First~~ Second attempt:



$$
\begin{aligned}
P_{\mathcal{G}+\mathcal{G}'} &: (X + X') \times (\Sigma \times \Sigma') \to (Y + Y') \\
P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\, x)\,(\sigma, \sigma') &= \mathsf{inl}\,(P_{\mathcal{G}}\, x\, \sigma) \\
P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inr}\, x)\,(\sigma, \sigma') &= \mathsf{inr}\,(P_{\mathcal{G}'}\, x\, \sigma') \\
&\vdots
\end{aligned}
$$

But: To define injections $\mathcal{G} \to \mathcal{G} + \mathcal{G}'$ we need a strategy component $\Sigma_G \to \Sigma_G \times \Sigma'_G$. ↯

Analysis

We kept both strategies around because we could not describe the situations when we needed one but not the other.

(This is reminiscent of implementing $A + B$ as $A \times B$, and supplying a dummy value as needed.)

Analysis

We kept both strategies around because we could not describe the situations when we needed one but not the other.

(This is reminiscent of implementing $A + B$ as $A \times B$, and supplying a dummy value as needed.)

But. . . what if we could be more precise about which strategy we need?

Open games in type theory

Introducing dependency

Old definition:

$$
\begin{aligned}
X &: \mathsf{Set} \\
S &: \mathsf{Set} \\
Y &: \mathsf{Set} \\
R &: \mathsf{Set} \\
\Sigma &: \mathsf{Set} \\
P &: X \to \Sigma \to Y \\
C &: X \to \Sigma \to R \to S \\
E &: X \to (Y \to R) \to \mathscr{P}(\Sigma)
\end{aligned}
$$

Introducing dependency

Dependently typed definition:

$$
\begin{aligned}
X &: \mathsf{Set} \\
S &: X \to \mathsf{Set} \\
Y &: \mathsf{Set} \\
R &: Y \to \mathsf{Set} \\
\Sigma &: X \to \mathsf{Set} \\
P &: (x : X) \to \Sigma\, x \to Y \\
C &: (x : X) \to (\sigma : \Sigma\, x) \to R\,(P\, x\, \sigma) \to S\, x \\
E &: (x : X) \to ((y : Y) \to R\, y) \to \mathscr{P}(\Sigma\, x)
\end{aligned}
$$

Introducing dependency

Dependently typed definition:

$$
\begin{aligned}
X &: \mathrm{Set} \\
S &: X \to \mathrm{Set} \\
Y &: \mathrm{Set} \\
R &: Y \to \mathrm{Set} \\
\Sigma &: X \to \mathrm{Set} \\
P &: (x : X) \to \Sigma\, x \to Y \\
C &: (x : X) \to (\sigma : \Sigma\, x) \to R\, (P\, x\, \sigma) \to S\, x \\
E &: (x : X) \to ((y : Y) \to R\, y) \to \mathscr{P}(\Sigma\, x)
\end{aligned}
$$

Note: (X, S) is a *container* [Abbott, Altenkirch, Ghani 2005].

Dependently typed open games

Let (X, S) and (Y, R) be containers.

Definition

A dependently typed open game $\mathcal{G} : (X, S) \to (Y, R)$ consists of:

- a family of sets $\Sigma_{\mathcal{G}} : X \to \mathrm{Set}$,
- a play function $P_{\mathcal{G}} : (x : X) \to \Sigma_{\mathcal{G}}(x) \to Y$,
- a coutility function
  $C_{\mathcal{G}} : (x : X) \to (\sigma : \Sigma_{\mathcal{G}}) \to R(P_{\mathcal{G}}\, x\, \sigma) \to S(x)$, and
- a equilibrium function
  $E_{\mathcal{G}} : (x : X) \to ((y : Y) \to R(y)) \to \mathscr{P}(\Sigma_{\mathcal{G}}(x))$.

Dependently typed open games

Let (X, S) and (Y, R) be containers.

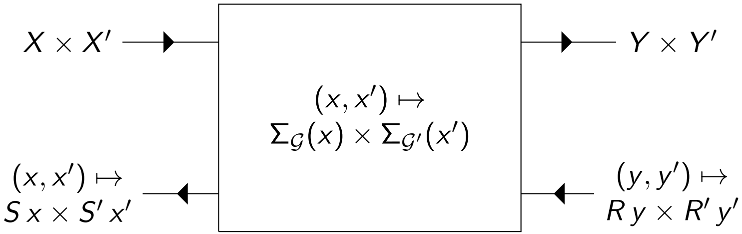
Definition

A dependently typed open game $\mathcal{G} : (X, S) \to (Y, R)$ consists of:

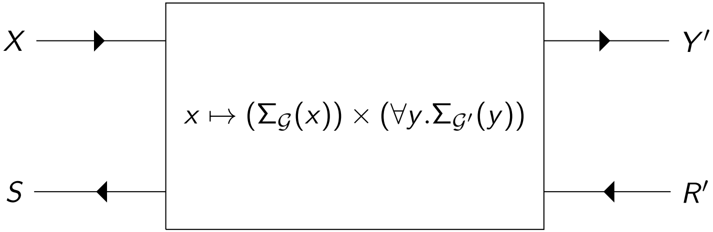
- a family of sets $\Sigma_\mathcal{G} : X \to \mathrm{Set}$,
- a play function $P_\mathcal{G} : (x : X) \to \Sigma_\mathcal{G}(x) \to Y$,
- a coutility function $C_\mathcal{G} : (x : X) \to (\sigma : \Sigma_\mathcal{G}) \to R(P_\mathcal{G}\, x\, \sigma) \to S(x)$, and
- a equilibrium function $E_\mathcal{G} : (x : X) \to ((y : Y) \to R(y)) \to \mathscr{P}(\Sigma_\mathcal{G}(x))$.

Observation: If S, R, $\Sigma_\mathcal{G}$ are constant families, this reduces to an ordinary open game.

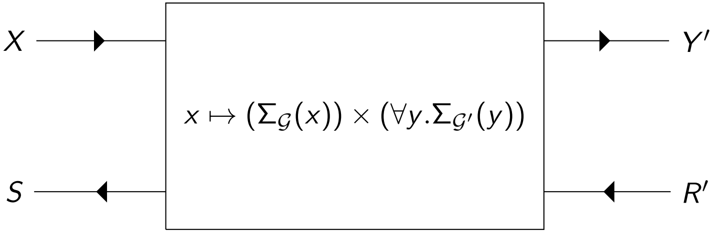
Parallel composition of dependently typed games



Sequential composition of dependently typed games



Sequential composition of dependently typed games



Note: "Alternative" definition

$$\Sigma_{\mathcal{G}' \circ \mathcal{G}}\, x = (\sigma : \Sigma_{\mathcal{G}}(x)) \times (\Sigma_{\mathcal{G}'}\, (P_{\mathcal{G}}\, x\, \sigma))$$

does not work.

Uniform function space $\forall y.B(y)$

Intuitively, consists of functions that make no computational use of their argument. (cf. “ghost variables” in Hoare logic).

Uniform function space $\forall y.B(y)$

Intuitively, consists of functions that make no computational use of their argument. (cf. "ghost variables" in Hoare logic).

Modelled by intersection in PER/realizability models.

Uniform function space $\forall y.B(y)$

Intuitively, consists of functions that make no computational use of their argument. (cf. "ghost variables" in Hoare logic).

Modelled by intersection in PER/realizability models.

In Agda: run-time irrelevance `@0` + Frobenius axiom

$$\forall x.(B \times P(x)) \cong B \times \forall x.P(x)$$

Symmetric monoidal structure

Theorem

(i) *The collection of containers, with open games* $\mathcal{G} : (X, S) \to (Y, R)$ *as morphisms, forms a symmetric monoidal category* DGame.

Symmetric monoidal structure

Theorem

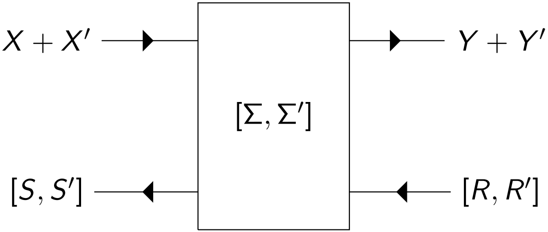
(i) *The collection of containers, with open games* $\mathcal{G} : (X, S) \to (Y, R)$ *as morphisms, forms a symmetric monoidal category* DGame*.*

(ii) *There is a identity-on-objects functor*

$$\iota : \mathrm{Cont} \to \mathrm{DGame}$$

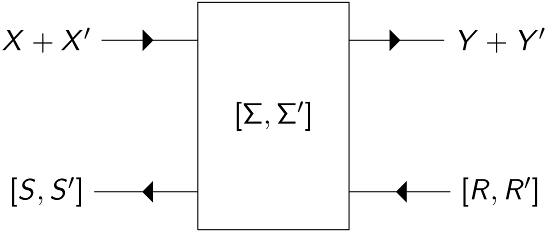
$\square$

Coproducts of dependently typed games



$$P_{\mathcal{G}+\mathcal{G}'}\,(\mathsf{inl}\,x, \sigma) = \mathsf{inl}\,(P_{\mathcal{G}}(x, \sigma))$$
$$P_{\mathcal{G}+\mathcal{G}'}\,(\mathsf{inr}\,x', \sigma') = \mathsf{inr}\,(P_{\mathcal{G}'}(x', \sigma'))$$

Coproducts of dependently typed games



$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inl}\, x, \sigma) = \mathsf{inl}\,(P_{\mathcal{G}}(x, \sigma))$$
$$P_{\mathcal{G}+\mathcal{G}'}(\mathsf{inr}\, x', \sigma') = \mathsf{inr}\,(P_{\mathcal{G}'}(x', \sigma'))$$

Also has the right universal property.

Summary

Compositional Game Theory in Type Theory

- Open games as a compositional model of game theory.
- Dependently typed open games for more precision in the model, and a mathematically nicer category of games (e.g. coproducts of games).

References

Jules Hedges
Towards compositional game theory
PhD thesis, Queen Mary University of London, 2016.

Neil Ghani, Jules Hedges, Viktor Winschel and Philipp Zahn
Compositional game theory
LICS 2018, pages 472–481, 2018.

Michael Abbott, Thorsten Altenkirch and Neil Ghani
Containers: constructing strictly positive types
Theoretical Computer Science 341 (1), pages 3–27, 2005.

Scottish Programming Languages and Verification Summer School

5–9 August 2019, Glasgow, Scotland

`http://www.macs.hw.ac.uk/splv/splv19/`

Chung-chieh Shan	Probabilistic programming
Phil Wadler	Programming Language Foundations in Agda
Neil Ghani	Category Theory
Conor McBride	Dependently Typed Programming
Ornela Dardha	Session types
Greg Michaelson, Rob Stewart	Domain-specific languages
Chris Brown	Parallel Programming