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.
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 $u : \Sigma \rightarrow \mathbb{R}^2$ 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

<table>
<thead>
<tr>
<th>Player 2</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,2)</td>
<td>(1,0)</td>
<td>(4,0)</td>
<td>(3,6)</td>
<td>(8,1)</td>
<td>(0,9)</td>
<td>(2,5)</td>
<td>(0,4)</td>
<td>(6,3)</td>
<td>(8,0)</td>
<td></td>
</tr>
<tr>
<td>(6,2)</td>
<td>(3,1)</td>
<td>(2,5)</td>
<td>(8,3)</td>
<td>(5,8)</td>
<td>(6,0)</td>
<td>(3,8)</td>
<td>(9,6)</td>
<td>(6,5)</td>
<td>(8,2)</td>
<td></td>
</tr>
<tr>
<td>(8,5)</td>
<td>(9,7)</td>
<td>(3,6)</td>
<td>(8,1)</td>
<td>(4,7)</td>
<td>(2,0)</td>
<td>(0,6)</td>
<td>(2,9)</td>
<td>(0,4)</td>
<td>(5,2)</td>
<td></td>
</tr>
<tr>
<td>(6,2)</td>
<td>(3,1)</td>
<td>(4,0)</td>
<td>(7,7)</td>
<td>(2,7)</td>
<td>(0,7)</td>
<td>(7,1)</td>
<td>(9,5)</td>
<td>(3,8)</td>
<td>(6,7)</td>
<td></td>
</tr>
<tr>
<td>(1,8)</td>
<td>(9,2)</td>
<td>(5,9)</td>
<td>(2,1)</td>
<td>(2,2)</td>
<td>(8,2)</td>
<td>(8,6)</td>
<td>(1,4)</td>
<td>(0,2)</td>
<td>(0,7)</td>
<td></td>
</tr>
<tr>
<td>(5,9)</td>
<td>(8,4)</td>
<td>(5,8)</td>
<td>(1,8)</td>
<td>(2,7)</td>
<td>(0,2)</td>
<td>(7,1)</td>
<td>(2,6)</td>
<td>(6,3)</td>
<td>(0,0)</td>
<td></td>
</tr>
<tr>
<td>(8,8)</td>
<td>(0,1)</td>
<td>(9,1)</td>
<td>(3,5)</td>
<td>(5,8)</td>
<td>(6,7)</td>
<td>(2,9)</td>
<td>(6,9)</td>
<td>(8,2)</td>
<td>(3,4)</td>
<td></td>
</tr>
<tr>
<td>(7,9)</td>
<td>(6,9)</td>
<td>(5,7)</td>
<td>(4,7)</td>
<td>(7,0)</td>
<td>(3,8)</td>
<td>(5,8)</td>
<td>(9,2)</td>
<td>(7,1)</td>
<td>(8,3)</td>
<td></td>
</tr>
<tr>
<td>(3,9)</td>
<td>(6,4)</td>
<td>(7,7)</td>
<td>(5,4)</td>
<td>(1,7)</td>
<td>(9,0)</td>
<td>(4,8)</td>
<td>(4,9)</td>
<td>(6,4)</td>
<td>(0,5)</td>
<td></td>
</tr>
<tr>
<td>(1,2)</td>
<td>(1,2)</td>
<td>(3,5)</td>
<td>(6,3)</td>
<td>(9,3)</td>
<td>(2,9)</td>
<td>(5,2)</td>
<td>(8,7)</td>
<td>(0,3)</td>
<td>(5,1)</td>
<td></td>
</tr>
<tr>
<td>(9,1)</td>
<td>(0,1)</td>
<td>(8,8)</td>
<td>(2,4)</td>
<td>(4,6)</td>
<td>(1,0)</td>
<td>(6,0)</td>
<td>(2,6)</td>
<td>(5,7)</td>
<td>(3,9)</td>
<td></td>
</tr>
</tbody>
</table>
The problem of scaling

<table>
<thead>
<tr>
<th>Player 1</th>
<th>Player 2</th>
<th>Player 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(6,8,0)</td>
<td>(9,7,0)</td>
<td>(9,4,0)</td>
</tr>
<tr>
<td>(4,1,0)</td>
<td>(8,7,0)</td>
<td>(3,5,0)</td>
</tr>
<tr>
<td>(2,3,0)</td>
<td>(7,1,7)</td>
<td>(1,5,0)</td>
</tr>
<tr>
<td>(5,6,0)</td>
<td>(1,8,0)</td>
<td>(4,9,0)</td>
</tr>
<tr>
<td>(7,1,0)</td>
<td>(7,1,7)</td>
<td>(4,3,0)</td>
</tr>
<tr>
<td>(1,7,0)</td>
<td>(6,7,0)</td>
<td>(1,4,1)</td>
</tr>
<tr>
<td>(1,2,2)</td>
<td>(2,1,0)</td>
<td>(2,9,3)</td>
</tr>
<tr>
<td>(3,6,4)</td>
<td>(6,7,6)</td>
<td>(1,3,0)</td>
</tr>
<tr>
<td>(8,4,0)</td>
<td>(4,3,3)</td>
<td>(5,3,7)</td>
</tr>
<tr>
<td>(9,9,6)</td>
<td>(8,7,6)</td>
<td>(1,4,0)</td>
</tr>
</tbody>
</table>

Diagram showing the problem of scaling with players and strategies.
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).
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

$X \in \text{Set}$ state of the game

$S \in \text{Set}$ coutility type

$Y \in \text{Set}$ moves of the game

$R \in \text{Set}$ utility type
Open games

Inside the box

Definition

An open game $G = (\Sigma_G, P_G, C_G, E_G) : (X, S) \rightarrow (Y, R)$ consists of:

- a set $\Sigma_G$ of strategy profiles,
- a play function $P_G : X \rightarrow \Sigma_G \rightarrow Y$,
- a coutility function $C_G : X \rightarrow \Sigma_G \rightarrow R \rightarrow S$, and
- an equilibrium function $E_G : X \rightarrow (Y \rightarrow R) \rightarrow P(\Sigma_G)$.
Open games
Inside the box

\[ \Sigma_G \]

\( X \rightarrow \Sigma_G \rightarrow Y \)

\( S \rightarrow \Sigma_G \rightarrow R \)

**Definition**

An open game \( G = (\Sigma_G, P_G, C_G, E_G) : (X, S) \rightarrow (Y, R) \) consists of:

- a set \( \Sigma_G \) of strategy profiles,
- a play function \( P_G : X \rightarrow \Sigma_G \rightarrow Y \),
- a coutility function \( C_G : X \rightarrow \Sigma_G \rightarrow R \rightarrow S \), and
- a equilibrium function \( E_G : X \rightarrow (Y \rightarrow R) \rightarrow \mathcal{P}(\Sigma_G) \).
Example: penalty shootout as an open game

\[ P(x, \sigma) = \sigma \]
\[ C(x, \sigma, r) = r \]
\[(a, b) \in E(x, k) \text{ iff } \pi_1(k(a, b)) \geq \pi_1(k(\bar{a}, b)) \text{ and } \pi_2(k(a, b)) \geq \pi_2(k(a, \bar{b})) \]
Parallel composition of open games

Proposition

The penalty shootout open game can be built as $P_1 \otimes P_2$, where $P_1, P_2 : (1, R) \rightarrow (\{L, R\}, R)$ with $\Sigma P_i = \{L, R\}$, and $a \in E P_i(x,k)$ iff $a \in \text{arg max}_{x \in \Sigma} \{k(x)\}$. 

8
Parallel composition of open games

\[
\begin{align*}
\Sigma & \quad \otimes \quad \Sigma' \\
S & \quad X \\
R & \quad Y \\
S' & \quad X' \\
R' & \quad Y' \\
S \times S' & \quad X \times X' \\
R \times R' & \quad Y \times Y' \\
\end{align*}
\]

Proposition

The penalty shootout open game can be built as \( P_1 \otimes P_2 \), where \( P_1, P_2 : (1, R) \rightarrow (\{L, R\}, R) \) with \( \Sigma P_i = \{L, R\} \), and \( a \in E P_i(x, k) \) iff \( a \in \arg \max_{x \in \Sigma} \{k(x)\} \).

8
Parallel composition of open games

![Diagram of parallel composition]

**Proposition**

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

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

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

8
Sequential composition

\[
\Sigma \times \Sigma' = X \downarrow Y \downarrow Z
\]
Sequential composition

\[ \Sigma \circ \Sigma' = \Sigma \times \Sigma' \]
Symmetric monoidal structure

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

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

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

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

(ii) There is an identity-on-objects functor

\[
i : \text{Set} \times \text{Set}^{\text{op}} \to \text{Game}
\]

with

\[
P_{i(f, g)}(x, \sigma) = f(x) \quad \quad C_{i(f, g)}(x, \sigma, r) = g(r).
\]
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:

\[
\begin{array}{c}
X + X' \rightarrow \\
\Sigma + \Sigma' \rightarrow \\
Y + Y' \\
S \times S' \rightarrow \\
R \times R' \\
\end{array}
\]

\[
P_{g+g'} : (X + X') \times (\Sigma + \Sigma') \rightarrow (Y + Y')
\]

\[
P_{g+g'}(\text{inl } x) (\text{inl } \sigma) = \{ ?_0 : Y + Y' \}
\]

\[
P_{g+g'}(\text{inl } x) (\text{inr } \sigma') = \{ ?_1 : Y + Y' \}
\]

\[
\vdots
\]
Coproduct construction attempts

First attempt:

\[
\begin{align*}
X + X' & \rightarrow \Sigma + \Sigma' \rightarrow Y + Y' \\
S \times S' & \rightarrow R \times R'
\end{align*}
\]

\[
P_{g+g'} : (X + X') \times (\Sigma + \Sigma') \rightarrow (Y + Y')
\]

\[
P_{g+g'}(\text{inl } x) \cdot (\text{inl } \sigma) = \text{inl } (P_g \times \sigma)
\]

\[
P_{g+g'}(\text{inl } x) \cdot (\text{inr } \sigma') = \{ ?_1 : Y + Y' \}
\]

\[
\vdots
\]
Coproduct construction attempts

First attempt:

\[
\begin{align*}
X + X' & \xrightarrow{\Sigma + \Sigma'} Y + Y' \\
S \times S' & \xrightarrow{R \times R'}
\end{align*}
\]

\[
P_{g+g'} : (X + X') \times (\Sigma + \Sigma') \rightarrow (Y + Y')
\]

\[
P_{g+g'}(\text{inl } x) (\text{inl } \sigma) = \text{inl} (P_g \times \sigma)
\]

\[
P_{g+g'}(\text{inl } x) (\text{inr } \sigma') = ??? \Downarrow
\]

::
Coprodct construction attempts

First Second attempt:

\[ X + X' \rightarrow Y + Y' \]

\[ S \times S' \rightarrow R \times R' \]

\[ \Sigma \times \Sigma' \]

\[ P_{g+g'} : (X + X') \times (\Sigma \times \Sigma') \rightarrow (Y + Y') \]

\[ P_{g+g'}(\text{inl } x) (\sigma, \sigma') = \{ ?_0 : Y + Y' \} \]

\[ P_{g+g'}(\text{inr } x) (\sigma, \sigma') = \{ ?_1 : Y + Y' \} \]

...
Coproduct construction attempts

First Second attempt:

\[ \begin{align*}
X + X' & \rightarrow \Sigma \times \Sigma' & Y + Y' \\
S \times S' & \rightarrow R \times R' \\
\end{align*} \]

\[ P_{g+g'} : (X + X') \times (\Sigma \times \Sigma') \rightarrow (Y + Y') \]

\[ P_{g+g'}(\text{inl } x) (\sigma, \sigma') = \text{inl } (P_g \times \sigma) \]

\[ P_{g+g'}(\text{inr } x) (\sigma, \sigma') = \{ ?_1 : Y + Y' \} \]

\[ \vdots \]
Coproduct construction attempts

First Second attempt:

\[ X + X' \rightarrow \Sigma \times \Sigma' \rightarrow Y + Y' \]

\[ S \times S' \rightarrow R \times R' \]

\[ P_{g+g'} : (X + X') \times (\Sigma \times \Sigma') \rightarrow (Y + Y') \]

\[ P_{g+g'}(\text{inl } x) (\sigma, \sigma') = \text{inl } (P_g \times \sigma) \]

\[ P_{g+g'}(\text{inr } x) (\sigma, \sigma') = \text{inr } (P_{g'} \times \sigma') \]

\[ \vdots \]

But: To define injections \( G \rightarrow G + G' \) we need a strategy component \( \Sigma_G \rightarrow \Sigma_G \times \Sigma'_G \).
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.)
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:

\[ X : \text{Set} \]
\[ S : \text{Set} \]
\[ Y : \text{Set} \]
\[ R : \text{Set} \]
\[ \Sigma : \text{Set} \]
\[ P : X \rightarrow \Sigma \rightarrow Y \]
\[ C : X \rightarrow \Sigma \rightarrow R \rightarrow S \]
\[ E : X \rightarrow (Y \rightarrow R) \rightarrow \mathcal{P}(\Sigma) \]
Introducing dependency

Dependently typed definition:

$X : \text{Set}$
$S : X \to \text{Set}$
$Y : \text{Set}$
$R : Y \to \text{Set}$
$\Sigma : X \to \text{Set}$
$P : (x : X) \to \Sigma x \to Y$
$C : (x : X) \to (\sigma : \Sigma x) \to R (P x \sigma) \to S x$
$E : (x : X) \to ((y : Y) \to R y) \to \mathcal{P} (\Sigma x)$
Introducing dependency

Dependently typed definition:

\[ X : \text{Set} \]
\[ S : X \rightarrow \text{Set} \]
\[ Y : \text{Set} \]
\[ R : Y \rightarrow \text{Set} \]
\[ \Sigma : X \rightarrow \text{Set} \]
\[ P : (x : X) \rightarrow \Sigma x \rightarrow Y \]
\[ C : (x : X) \rightarrow (\sigma : \Sigma x) \rightarrow R (P \times \sigma) \rightarrow S x \]
\[ E : (x : X) \rightarrow ((y : Y) \rightarrow R y) \rightarrow \mathcal{P} (\Sigma x) \]

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 \(G: (X, S) \rightarrow (Y, R)\) consists of:

- a family of sets \(\Sigma_G: X \rightarrow \text{Set},\)
- a play function \(P_G: (x : X) \rightarrow \Sigma_G(x) \rightarrow Y,\)
- a coutility function \(C_G: (x : X) \rightarrow (\sigma : \Sigma_G) \rightarrow R(P_G \times \sigma) \rightarrow S(x),\) and
- a equilibrium function \(E_G: (x : X) \rightarrow ((y : Y) \rightarrow R(y)) \rightarrow P(\Sigma_G(x)).\)
Dependently typed open games

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

**Definition**
A dependently typed open game \(G : (X, S) \rightarrow (Y, R)\) consists of:

- a family of sets \(\Sigma_G : X \rightarrow \text{Set},\)
- a play function \(P_G : (x : X) \rightarrow \Sigma_G(x) \rightarrow Y,\)
- a coutility function 
  \(C_G : (x : X) \rightarrow (\sigma : \Sigma_G) \rightarrow R(P_G \times \sigma) \rightarrow S(x),\) and
- a equilibrium function 
  \(E_G : (x : X) \rightarrow ((y : Y) \rightarrow R(y)) \rightarrow \mathcal{P}(\Sigma_G(x)).\)

**Observation:** If \(S, R, \Sigma_G\) are constant families, this reduces to an ordinary open open game.
Parallel composition of dependently typed games

\[ X \times X' \mapsto \Sigma_{g}(x) \times \Sigma_{g'}(x') \]

\[ (x, x') \mapsto (y, y') \mapsto R y \times R' y' \]
Sequential composition of dependently typed games

\[ x \mapsto (\Sigma_G(x)) \times (\forall y. \Sigma_{G'}(y)) \]

Note: "Alternative" definition

\[ \Sigma_G \circ G_x = (\sigma: \Sigma_G(x)) \times (\Sigma_{G'}(P_{G}x \sigma)) \]

does not work.
Sequential composition of dependently typed games

\[
X \xrightarrow{x \mapsto (\Sigma G(x)) \times (\forall y.\Sigma G'(y))} Y' \\
S \xrightarrow{} R'
\]

Note: “Alternative” definition

\[
\Sigma_{G' \circ G} x = (\sigma : \Sigma G(x)) \times (\Sigma G' (P_G \times \sigma))
\]

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 $\mathbb{0} +$ 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 $\text{DGame}$. 
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 \( \text{DGame} \).

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

\[ \iota : \text{Cont} \to \text{DGame} \]
Coproducts of dependently typed games

\[ X + X' \xrightarrow{\quad} [\Sigma, \Sigma'] \xleftarrow{\quad} Y + Y' \]

\[ [S, S'] \xrightarrow{\quad} [R, R'] \]

\[ P_{g + g'} (\text{inl } x, \sigma) = \text{inl} (P_g (x, \sigma)) \]

\[ P_{g + g'} (\text{inr } x', \sigma') = \text{inr} (P_{g'} (x', \sigma')) \]
Coproducts of dependently typed games

\[ X + X' \rightarrow [\Sigma, \Sigma'] \rightarrow Y + Y' \]

\[ [S, S'] \rightarrow [R, R'] \]

\[ P_{G+G'} (\text{inl } x, \sigma) = \text{inl } (P_G(x, \sigma)) \]
\[ P_{G+G'} (\text{inr } x', \sigma') = \text{inr } (P_{G'}(x', \sigma')) \]

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).
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

Michael Abbott, Thorsten Altenkirch and Neil Ghani
Containers: constructing strictly positive types
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: Dependenly Typed Programming
Ornela Dardha: Session types
Greg Michaelson, Rob Stewart: Domain-specific languages
Chris Brown: Parallel Programming