Higher-Order Game Theory

Paulo Oliva
Queen Mary University of London

TACL 2015
23 June, Ischia, Italy

Joint work with…

Martin
Escardo

Jules
Hedges

Evguenia
Sprits

Philipp
Zahn

Viktor
Winschel

Plan

1. Players
2. Games
3. Equilibria
4. Monads

Running Example

A Simple Game

- Two contestants {A, B}
- Three judges $\{J_1, J_2, J_3\}$
- Judge J_1 prefers $A > B$
- Judge J_2 prefers $B > A$
- Judge J_3 wants to vote for the winner

Matrix Representation

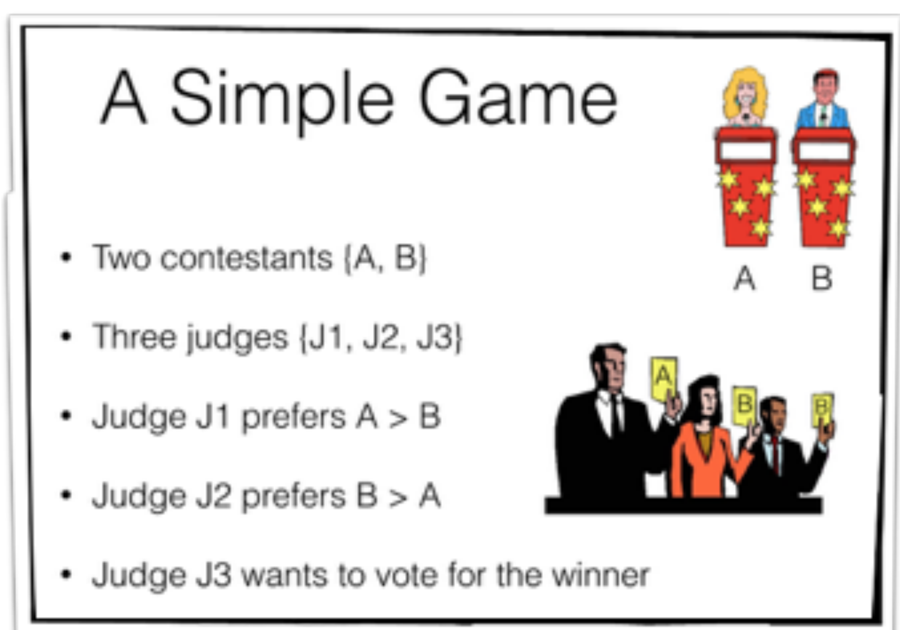
J_1 J_2 \ J_3	A	B
AA	1,0,1	1,0,0
AB	1,0,1	0,1,1
BA	1,0,1	0,1,1
BB	0,1,0	0,1,1

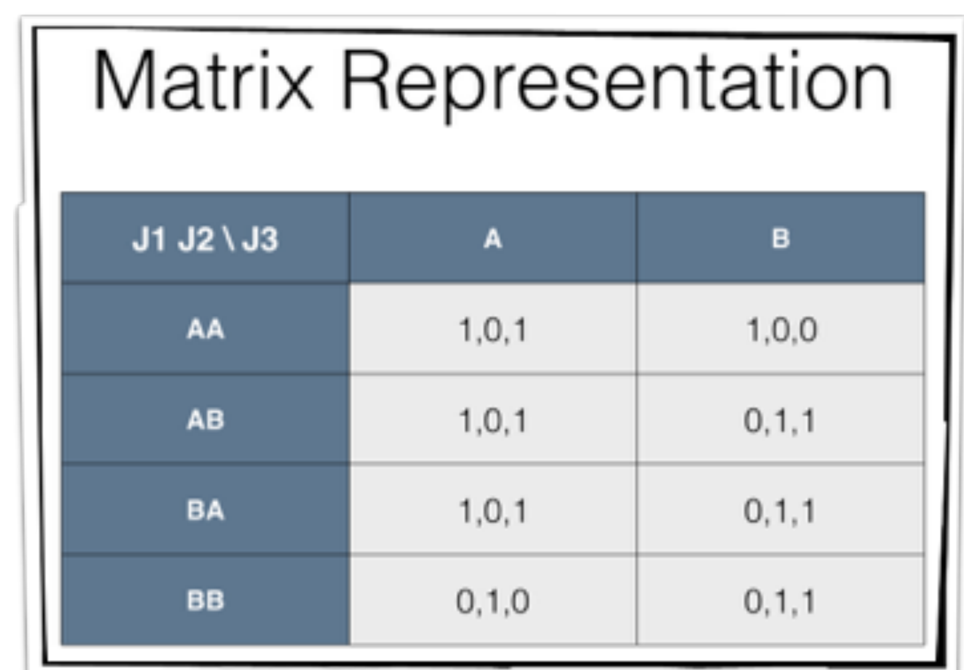
Five Judges

J_1 J_2 J_3 \ J_4 J_5	AA	AB	BA	BB
AAA	1,1,0,1,1	1,1,0,1,1	1,1,0,0,1	1,1,0,0,1
AAB	1,1,0,1,1	1,1,0,1,1	1,1,0,0,1	0,0,1,1,0
ABA	1,0,0,1,1	1,0,0,1,1	1,0,0,0,1	0,1,1,1,0
ABB	1,0,0,1,1	0,1,1,0,0	0,1,1,1,0	0,1,1,1,0
BAA	1,1,0,1,1	1,1,0,1,1	1,1,0,0,1	0,0,1,1,0
BAB	1,1,0,1,1	0,0,1,0,0	0,0,1,1,0	0,0,1,1,0
BBA	1,0,0,1,1	0,1,1,0,0	0,1,1,1,0	0,1,1,1,0
BBB	0,1,1,0,0	0,1,1,0,0	0,1,1,1,0	0,1,1,1,0

Representation vs Model

- Normal-form matrix **representations** are good to calculate properties of games, e.g. equilibria
- Not so good for **modelling** the ‘goals’ of players





Matrix Representation

J1 J2 \ J3	A	B
AA	1,0,1	1,0,0
AB	1,0,1	0,1,1
BA	1,0,1	0,1,1
BB	0,1,0	0,1,1

Modelling Language

- **Formal** (precise and subject to manipulation)
- **Expressive** (can capture different ‘situations’)
- **Faithful** (captures precisely the game)
- **High level** (we can understand)
- **Modular** (whole built of individual parts)

Modelling Players

Player Context

- If judges 1 and 2 fix their moves, say A and B, that defines a **context** for judge 3
- If judge 3 chooses A then A wins
- If judge 3 chooses B then B wins
- Context = a function from moves to outcomes

Player Context

- Assume a player is choosing moves in X having in mind an outcome in R
- This player's contexts are functions $f : X \longrightarrow R$
- When all other opponents have fixed their moves, this defines a context for the player
- **Note**: In a particular game, for particular opponents, some contexts might not arise

Player Context

J1 J2 \ J3	A	B
AA	1,0,1 [A]	1,0,0 [A]
AB	1,0,1 [A]	0,1,1 [B]
BA	1,0,1 [A]	0,1,1 [B]
BB	0,1,0 [B]	0,1,1 [B]

- In this game there are **three** possible contexts for judge 3 (which are they?)

Player

- Assume players are choosing moves in X having in mind an outcome in R

- Players will be modelled as mappings from **contexts** to **good moves**

$$(X \longrightarrow R) \longrightarrow P(X)$$

- Slogan: *To know a player is to know his optimal moves in any possible context*

Our Three Judges

- $X = R = \{A, B\}$
- Judge 1 is $\text{argmax} : (X \longrightarrow R) \longrightarrow P(X)$ with respect to the ordering $A > B$
- Judge 2 is $\text{argmax} : (X \longrightarrow R) \longrightarrow P(X)$ with respect to the ordering $B > A$
- Judge 3 is $\text{fix} : (X \longrightarrow R) \longrightarrow P(X)$

$$\text{fix}(p) = \{\ x : p(x) = x\ \}$$

```haskell
type Player r x = (x -> r) -> [x]
data Cand = A | B deriving (Eq,Ord,Enum,Show)
type Judge x = Player Cand x

cand = enumFrom A  -- List of candidates [A, B,..]

-- Judge that prefer A > B
argmax1 :: Judge Cand
argmax1 p = [ x | x <- cand, p x == minimum (map p cand) ]

-- Judge that prefer B > A
argmax2 :: Judge Cand
argmax2 p = [ x | x <- cand, p x == maximum (map p cand) ]

-- Judge that wants to vote for the winner
fix :: Judge Cand
fix p = [ x | x <- cand, p x == x ]
```

Implementing in Haskell

Our Three Judges

- Shouldn’t Judge 1 be the constant mapping

$$J_1(p) = \{ A \}$$

- Shouldn’t Judge 2 be the constant mapping

$$J_2(p) = \{ B \}$$

- No! We are defining the player irrespective of the concrete context, which includes the game itself!!

Modelling Games

The Outcome Function

- Outcome function = map from moves to outcome

$$X_1 \times \ldots \times X_n \longrightarrow R$$

- Suppose we change the rules of the game so that the candidate with least votes wins
  - ✳ If J_1 wants A to win he better vote for B
  - ✳ If J_2 wants B to win he better vote for A
  - ✳ No change to selection function representation!

Higher-order Game

- Number of players: n
- Types: moves $(X_1, \ldots, X_n)$ and outcome (R)
- Selection functions for each player $i = 1 \ldots n$

$$\varepsilon_i \;:\; (X_i \longrightarrow R) \longrightarrow P(X_i)$$

- An outcome function

$$q \;:\; X_1 \times \ldots \times X_n \longrightarrow R$$

Example 1

- Number of players: 3
- $X_1 = X_1 = X_3 = R = \{ A , B \}$
- Player 1, $\text{argmax} : (X_1 \longrightarrow R) \longrightarrow P(X_1)$, with $A > B$
- Player 2, $\text{argmax} : (X_2 \longrightarrow R) \longrightarrow P(X_2)$, with $B > A$
- Player 3, $\text{fix} : (X_3 \longrightarrow R) \longrightarrow P(X_3)$
- $q(x_1, x_2, x_3) = \text{majority}(x_1, x_2, x_3)$

Example 2

- Number of players: 5
- $X_1 = X_1 = X_3 = X_4 = X_5 = R = \{ A , B \}$
- Player 1 and 5 are argmax, with $A > B$
- Player 3 is argmax, with $B > A$
- Player 2 and 4 are fix
- $q(x_1, x_2, x_3, x_4, x_5) = \text{majority}(x_1, x_2, x_3, x_4, x_5)$

Modelling Language

- **Formal** (precise and subject to manipulation) ✔
- **Expressive** (can capture different ‘situations’) ✔
- **Faithful** (captures precisely the game) ✔
- **High level** (we can understand) ✔
- **Modular** (whole built of individual parts) ✔

Aggregate Preferences

- *Judge X wants A to win, if possible. Otherwise, he would rather vote with the winner.*

$$\varepsilon^{X}(p) = \text{if } A \in \mathrm{Img}(p) \text{ then } p^{-1}(\{A\}) \text{ else } \mathrm{fix}(p)$$

- *Judge Y is happy if either the best or worse candidate wins.*

$$\varepsilon^{Y}(p) = \mathrm{argmax}(p) \cup \mathrm{argmin}(p)$$

Modelling
Equilibrium Concepts

Equilibrium Strategies

- Judge J_1 prefers $A > B$
- Judge J_2 prefers $B > A$
- Judge J_3 wants to vote for the winner

J_1 J_2 \ J_3	A	B
AA	1,0,1	1,0,0
AB	1,0,1	0,1,1
BA	1,0,1	0,1,1
BB	0,1,0	0,1,1

(Classic) Nash Equilibrium

- Let the payoff function of player i be

  $q_i : X_1 \times \ldots \times X_n \longrightarrow \text{Real}$

- A choice of moves is in **equilibrium** if no player has an incentive to deviate from his/her choice

- Player i has **no incentive to deviate** if

  $q_i(x_1,\ldots,x_n) \geq q_i(x_1,\ldots,y,\ldots,x_n)$, for all y in X_i

Five Judges

J_1 J_2 J_3 \ J_4 J_5	AA	AB	BA	BB
AAA	1,1,0,1,1	1,1,0,1,1	1,1,0,0,1	1,1,0,0,1
AAB	1,1,0,1,1	1,1,0,1,1	1,1,0,0,1	0,0,1,1,0
ABA	1,0,0,1,1	1,0,0,1,1	1,0,0,0,1	0,1,1,1,0
BAA	1,1,0,1,1	1,1,0,1,1	1,1,0,0,1	0,0,1,1,0
ABB	1,0,0,1,1	0,1,1,0,0	0,1,1,1,0	0,1,1,1,0
BAB	1,1,0,1,1	0,0,1,0,0	0,0,1,1,0	0,0,1,1,0
BBA	1,0,0,1,1	0,1,1,0,0	0,1,1,1,0	0,1,1,1,0
BBB	0,1,1,0,0	0,1,1,0,0	0,1,1,1,0	0,1,1,1,0

Nash Going High

- Player i has **no incentive to deviate** if

  $q_i(x_1,\ldots,x_n) \geq q_i(x_1,\ldots,y,\ldots,x_n)$, for all $y \in X_i$

- Equivalent to

  $x_i \in \mathrm{argmax}\ (\lambda y.q_i(x_1,\ldots,y,\ldots,x_n))$

- (Higher-order) player i has no incentive to deviate if

  $x_i \in \varepsilon_i\ (\lambda y.q(x_1,\ldots,y,\ldots,x_n))$

Equilibrium Checker

```haskell
-- Unilateral context
cont :: ([Cand] -> Cand) -> [Cand] -> Int -> Cand -> Cand
cont q xs i x = q $ (take i xs) ++ [x] ++ (drop (i+1) xs)

-- Equilibrium checking = Global player
global :: [Judge Cand] -> Judge [Cand]
global js q = [ xs | xs <- plays,
                     all (good xs) (zip [0..] js) ]
  where
     n = length js
     plays = sequence (replicate n cand)
     good xs (i,e) = elem (xs !! i) (e (cont q xs i))
```

Monads

Player’s Strategy

- **Player’s description**

$$(X \longrightarrow R) \longrightarrow P(X)$$

- **Player’s strategy**

$$(X \longrightarrow R) \longrightarrow X$$

Monads

DEFINITION 1.2 (Strong monad). *Let T be a meta-level unary operation on simple types, that we will call a* type operator. *A type operator T is called a* strong monad *if we have a family of closed terms*

$$\begin{aligned} \eta_X &: X \to TX \\ (\cdot)^\dagger &: (X \to TY) \to (TX \to TY) \end{aligned}$$

satisfying the laws

(*i*) $(\eta_X)^\dagger = \mathrm{id}_{TX}$

(*ii*) $g^\dagger \circ \eta_Y = g$

(*iii*) $(g^\dagger \circ f)^\dagger = g^\dagger \circ f^\dagger$

where $g \colon Y \to TR$ and $f \colon X \to TY$.

Selection Monad

- Fix R. The type mapping

$$J\,X = (X \longrightarrow R) \longrightarrow X$$

is a **strong monad**

```
data J r x = J { selection :: (x -> r) -> x }

monJ :: J r x -> (x -> J r y) -> J r y
monJ e f = J (\p -> b p (a p))
   where
      a p = selection e $ (\x -> p (b p x))
      b p x = selection (f x) p

instance Monad (J r) where
   return x = J(\p -> x)
   e >>= f = monJ e f
```

Product of Selection Functions

- Strong monads support two operations

$$(T\ X) \times (T\ Y) \longrightarrow T\ (X \times Y)$$

- So we have two "products" of type

$$(J\ X) \times (J\ Y) \longrightarrow J\ (X \times Y)$$

- **Game theoretic interpretation**:
  A way of combining players' strategies!

Iterated Product

`sequence :: Monad m => [m a] -> m [a]`

base Prelude, base Control.Monad

Evaluate each action in the sequence from left to right, and collect the results.

- One product $(J\ X) \times (J\ Y) \longrightarrow J\ (X \times Y)$ can be iterated

$$\Pi_i\ J\ X_i \longrightarrow J\ \Pi_i\ X_i$$

- Backward induction: Calculates sub-game perfect equilibria of sequential games (Escardó/O'2012)

Where all this came from…

Topology

- Theorem[Tychonoff].
  Countable product of compact sets is compact

- **Searchable sets** = sets + selection function

$$(X \longrightarrow \mathrm{Bool}) \longrightarrow X$$

- **Searchable sets** = compact sets

- Theorem[Escardo].
  Countable product of searchable sets is searchable

  Proof. Countable product of selection functions

Logic

- T = Gödel's calculus of primitive recursive functionals
- Bar recursion BR: Spector (1962) computational interpretation of countable choice
- Interpretation of classical analysis into T + BR
- Theorem[Escardó/O.'2014] BR is T-equivalent to iterated product of selection function

Categories & Algebras

- Given any strong monad T and a T-algebra R then

$$J^T X = (X \longrightarrow R) \longrightarrow T X$$

  is also a **strong monad**

- Currently playing with different T's

  1. (finite) power-set monad (*Herbrand interpretation*)
  2. distribution monad (*mixed strategies*)

References

- Escardó and Oliva. *Selection functions, bar recursion and backward induction*. Mathematical Structures in Computer Science, 20(2):127-168, 2010

- Escardó and Oliva. *Sequential games and optimal strategies*. Proceedings of the Royal Society A, 467:1519-1545, 2011

- Hedges, Oliva, Sprits, Zahn, and Winschel. *A higher-order framework for decision problems and games*, ArXiv, http://arxiv.org/abs/1409.7411, 2014