Nash Equilibria and Unbounded Games

Paulo Oliva
Queen Mary University of London

CiE 2015
30 June, Bucharest
Joint work with…

Martin Escardo
Jules Hedges
Evguenia Sprits
Philipp Zahn
Viktor Winschel
Plan

1. Players
2. Simultaneous Games
3. Equilibria
4. (Infinite) Sequential Games
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

<table>
<thead>
<tr>
<th>( J_1 ) J_2 ( J_3 )</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>1,0,1</td>
<td>1,0,0</td>
</tr>
<tr>
<td>AB</td>
<td>1,0,1</td>
<td>0,1,1</td>
</tr>
<tr>
<td>BA</td>
<td>1,0,1</td>
<td>0,1,1</td>
</tr>
<tr>
<td>BB</td>
<td>0,1,0</td>
<td>0,1,1</td>
</tr>
</tbody>
</table>
# Five Judges

<table>
<thead>
<tr>
<th>J₁ J₂ J₃ \ J₄ J₅</th>
<th>AA</th>
<th>AB</th>
<th>BA</th>
<th>BB</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>1,1,0,1,1</td>
<td>1,1,0,1,1</td>
<td>1,1,0,0,1</td>
<td>1,1,0,0,1</td>
</tr>
<tr>
<td>AAB</td>
<td>1,1,0,1,1</td>
<td>1,1,0,1,1</td>
<td>1,1,0,0,1</td>
<td>0,0,1,1,0</td>
</tr>
<tr>
<td>ABA</td>
<td>1,0,0,1,1</td>
<td>1,0,0,1,1</td>
<td>1,0,0,0,1</td>
<td>0,1,1,1,0</td>
</tr>
<tr>
<td>ABB</td>
<td>1,0,0,1,1</td>
<td>0,1,1,0,0</td>
<td>0,1,1,1,0</td>
<td>0,1,1,1,0</td>
</tr>
<tr>
<td>BAA</td>
<td>1,1,0,1,1</td>
<td>1,1,0,1,1</td>
<td>1,1,0,0,1</td>
<td>0,0,1,1,0</td>
</tr>
<tr>
<td>BAB</td>
<td>1,1,0,1,1</td>
<td>0,0,1,0,0</td>
<td>0,0,1,1,0</td>
<td>0,0,1,1,0</td>
</tr>
<tr>
<td>BBA</td>
<td>1,0,0,1,1</td>
<td>0,1,1,0,0</td>
<td>0,1,1,1,0</td>
<td>0,1,1,1,0</td>
</tr>
<tr>
<td>BBB</td>
<td>0,1,1,0,0</td>
<td>0,1,1,0,0</td>
<td>0,1,1,1,0</td>
<td>0,1,1,1,0</td>
</tr>
</tbody>
</table>
Representation vs Model

• Normal-form matrix representations are good for calculating properties of games, e.g. equilibria

• Not so good for modelling the ‘goals’ of players
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
Concrete Context

• Assume rules of the game are fixed

• If judges 1 and 2 fix their moves, say A and B, that defines a **concrete context** for judge 3

• If judge 3 chooses A then A wins

• If judge 3 chooses B then B wins
Abstract Context

• Assume a player is choosing moves in $X$ having in mind an outcome in $R$

• **Abstract contexts** are functions $f : X \rightarrow R$

• Every concrete context determines an abstract one
Abstract vs Concrete

- **Note**: In a particular game, for particular opponents, some abstract contexts might not arise

<table>
<thead>
<tr>
<th>J1</th>
<th>J2 \ J3</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>1,0,1</td>
<td>1,0,0</td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>1,0,1</td>
<td>0,1,1</td>
<td></td>
</tr>
<tr>
<td>BA</td>
<td>1,0,1</td>
<td>0,1,1</td>
<td></td>
</tr>
<tr>
<td>BB</td>
<td>0,1,0</td>
<td>0,1,1</td>
<td></td>
</tr>
</tbody>
</table>

- In this game there are **three** abstract contexts for judge 3 (but **four** concrete ones)
Player

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

• Players will be modelled as mappings from abstract contexts to good moves

$(X \rightarrow R) \rightarrow P(X)$

• Slogan: To know a player is to know his optimal moves in any possible abstract context
Our Three Judges

- $X = R = \{A, B\}$

- Judge 1 is $\text{argmax} : (X \rightarrow R) \rightarrow P(X)$ with respect to the ordering $A > B$

- Judge 2 is $\text{argmax} : (X \rightarrow R) \rightarrow P(X)$ with respect to the ordering $B > A$

- Judge 3 is $\text{fix} : (X \rightarrow R) \rightarrow P(X)$

\[\text{fix}(p) = \{ x : p(x) = x \}\]
Implementing in 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 ]
Simultaneous Games
The Outcome Function

• Outcome function = map from moves to outcome

\[ X_1 \times \ldots \times X_n \rightarrow R \]

• 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 $i = 1 \ldots n$
  
  $\varepsilon_i : (X_i \rightarrow R) \rightarrow P(X_i)$

- **An outcome function**

  $q : X_1 \times \ldots \times X_n \rightarrow R$
Example 1

- Number of players: 3

- $X_1 = X_1 = X_3 = R = \{ A, B \}$

- Player 1, $\text{argmax} : (X_1 \rightarrow R) \rightarrow P(X_1)$, with $A > B$

- Player 2, $\text{argmax} : (X_2 \rightarrow R) \rightarrow P(X_2)$, with $B > A$

- Player 3, $\text{fix} : (X_3 \rightarrow R) \rightarrow P(X_3)$

- $q(x_1, x_2, x_3) = \text{majority}(x_1, x_2, x_3)$
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) ✔
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

<table>
<thead>
<tr>
<th>$J_1 \ J_2 \ \backslash \ J_3$</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>1,0,1</td>
<td>1,0,0</td>
</tr>
<tr>
<td>AB</td>
<td>1,0,1</td>
<td>0,1,1</td>
</tr>
<tr>
<td>BA</td>
<td>1,0,1</td>
<td>0,1,1</td>
</tr>
<tr>
<td>BB</td>
<td>0,1,0</td>
<td>0,1,1</td>
</tr>
</tbody>
</table>
(Classic) Nash Equilibrium

• Let the payoff function of player i be

\[ q_i : X_1 \times \ldots \times X_n \rightarrow \text{Real} \]

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

• Player i has \textbf{no incentive to deviate} if

\[ q_i(x_1,\ldots,x_n) \geq q_i(x_1,\ldots,y,\ldots,x_n), \text{ for all } y \text{ in } X_i \]
Nash Going High

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

\[ q_i(x_1, \ldots, x_n) \geq q_i(x_1, \ldots, y, \ldots, x_n) \text{, for all } y \in X_i \]

• Equivalent to

\[ x_i \in \text{argmax } (\lambda y. q_i(x_1, \ldots, y, \ldots, x_n)) \]

• (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

-- Unilateral context

\[
\text{cont} \::\: ([\text{Cand}] \rightarrow \text{Cand}) \rightarrow ([\text{Cand}] \rightarrow \text{Int} \rightarrow \text{Cand} \rightarrow \text{Cand})
\]
\[
\text{cont} \ q \ \text{xs} \ i \ x = q \ \cdot\ (\text{take} \ i \ \text{xs}) \ \cdot\ \text{[x]} \ \cdot\ \text{++} \ \cdot\ \text{([drop \ (i+1) \ \text{xs}])}
\]

-- Equilibrium checking = Global player

\[
\text{global} \ ::\: \text{[[Judge \ Cand]]} \rightarrow \text{Judge \ [Cand]}
\]
\[
\text{global} \ js \ q = [ \ \text{xs} \mid \text{xs} \leftarrow \text{plays},
\quad \text{all} \ (\text{good} \ \text{xs}) \ (\text{zip} \ [0..] \ \text{js}) ]
\]

\text{where}

\[
\text{n} = \text{length} \ \text{js}
\]
\[
\text{plays} = \text{sequence} \ (\text{replicate} \ \text{n} \ \text{cand})
\]
\[
\text{good} \ \text{xs} \ (i,e) = \text{elem} \ (\text{xs} \ !\! \ i) \ (e \ (\text{cont} \ q \ \text{xs} \ i))
\]
Sequential Games
Player’s Strategy

• Player’s description

\[(X \rightarrow R) \rightarrow P(X)\]

• Player’s strategy

\[(X \rightarrow R) \rightarrow X\]
Selection Monad

- Fix $R$. The type mapping

$$J X = (X \rightarrow R) \rightarrow X$$

is a **strong monad**

```haskell
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

\[(T X) \times (T Y) \rightarrow T (X \times Y)\]

• So we have two “products” of type

\[(J X) \times (J Y) \rightarrow J (X \times Y)\]

• **Game theoretic interpretation**: Sequentially combining players’ strategies!
Iterated Product

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

\[ \prod_i J X_i \rightarrow J \prod_i X_i \]

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

- Escardó and Oliva
  *Selection functions, bar recursion and backward induction.*

- Escardó and Oliva
  *Sequential games and optimal strategies.*

- Escardó and Oliva
  *Computing Nash equilibria of unbounded games*

- Hedges, Oliva, Sprits, Zahn, and Winschel
  *A higher-order framework for decision problems and games*