An Existence Theorem of Nash Equilibrium in Coq and Isabelle

Stéphane Le Roux, Érik Martin-Dorel and Jan-Georg Smaus

GandALF 2017

20 September 2017

A game theory result formalised in Coq and Isabelle

- Le Roux has shown a result on two-player games: starting from a game with multiple outcomes, one can derive a game that maps those outcomes into just two possible outcomes, namely that player 1 wins or player 2 wins.

A game theory result formalised in Coq and Isabelle

- Le Roux has shown a result on two-player games: starting from a game with multiple outcomes, one can derive a game that maps those outcomes into just two possible outcomes, namely that player 1 wins or player 2 wins.
- If the game is such that any way of deriving such a win-lose game leads to a game with a Nash equilibrium (and hence a pre-determined winner), then the original game also has a Nash equilibrium.
- We prove this result in Coq and Isabelle.

Game forms

Definition

A game form is a tuple $\langle A, (S_a)_{a \in A}, O, v \rangle$ such that

- A is a non-empty set of players,
- $\prod_{a \in A} S_a$ is a non-empty Cartesian product of strategy profiles, where S_a represents the strategies available to player a,
- O is a non-empty set of possible outcomes,
- $v : \prod_{a \in A} S_a \to O$ is the outcome function.

Providing $\prec_a$, a binary preference relation over O for each player a, constitutes a game.

Nash equilibrium

Definition

Let $g = \langle A, (S_a)_{a \in A}, O, v, (\prec_a)_{a \in A} \rangle$ be a game. A strategy profile s in $S := \prod_{a \in A} S_a$ is a Nash equilibrium if it makes every player a stable, *i.e.* $v(s) \not\prec_a v(s')$ for all $s' \in S$ that differ from s at most at the a-component.

$$\mathrm{NE}(s) := \forall a \in A, \forall s' \in S, \quad (\forall b \in A \setminus \{a\},\, s_b = s'_b) \;\Rightarrow\; v(s) \not\prec_a v(s')$$

Four games

Nash equilibria

	2_l	2_r
1_t	1, 0	**5, 0**
1_b	**2, 4**	5, 3

	2_l	2_r
1_t	0, 1	1, 0
1_b	1, 0	0, 1

	2_l	2_r
1_t	**2, 1**	0, 0
1_b	0, 0	**1, 2**

	2_l	2_r
1_t	0, 1	0, 1
1_b	**1, 0**	**1, 0**

Four games

Nash equilibria

	2_l	2_r
1_t	1, 0	**5, 0**
1_b	**2, 4**	5, 3

	2_l	2_r
1_t	0, 1	1, 0
1_b	1, 0	0, 1

	2_l	2_r
1_t	**2, 1**	0, 0
1_b	0, 0	**1, 2**

	2_l	2_r
1_t	0, 1	0, 1
1_b	**1, 0**	**1, 0**

Four games

Nash equilibria

$(1_t, 2_r)$ is an NE

	2_l	2_r
1_t	1, 0	**5, 0**
1_b	**2, 4**	5, 3

	2_l	2_r
1_t	0, 1	1, 0
1_b	1, 0	0, 1

	2_l	2_r
1_t	**2, 1**	0, 0
1_b	0, 0	**1, 2**

	2_l	2_r
1_t	0, 1	0, 1
1_b	**1, 0**	**1, 0**

We will concentrate on two-player games from now on.

Win-lose games

Definition

- A win-lose game is a game where $A = \{1, 2\}$ and $O = \{(1, 0), (0, 1)\}$ with preferences as expected ...
- Winning strategy $s_1 \in S_1$ for Player 1: $v(s_1, s_2) = (1, 0)$ for all $s_2 \in S_2$. Analogous for Player 2.
- A win-lose game such that one player has a winning strategy is said to be determined.

The four games again

	2_l	2_r
1_t	1, 0	**5, 0**
1_b	**2, 4**	5, 3

	2_l	2_r
1_t	0, 1	1, 0
1_b	1, 0	0, 1

Non-determined win-lose

	2_l	2_r
1_t	**2, 1**	0, 0
1_b	0, 0	**1, 2**

	2_l	2_r
1_t	0, 1	0, 1
1_b	**1, 0**	**1, 0**

Win-lose with winning strategy 1_b

The four games again

	2_l	2_r
1_t	1, 0	**5, 0**
1_b	**2, 4**	5, 3

	2_l	2_r
1_t	0, 1	1, 0
1_b	1, 0	0, 1

Non-determined win-lose

	2_l	2_r
1_t	**2, 1**	0, 0
1_b	0, 0	**1, 2**

	2_l	2_r
1_t	0, 1	0, 1
1_b	**1, 0**	**1, 0**

Win-lose with winning strategy 1_b

We will concentrate on two-player games from now on.

Derived games

Definition

Let $gf = \langle\{1,2\}, S_1, S_2, O, v\rangle$ be a two-player game form.

- For all $\prec_1, \prec_2 \subseteq O^2$ the game $\langle\{1,2\}, S_1, S_2, O, v, \{\prec_1, \prec_2\}\rangle$ is said to be derived from gf.
- Let wl be a function from O to $\{(1,0),(0,1)\}$. The win-lose game $\langle S_1, S_2, wl \circ v\rangle$ is also said to be derived from gf.
- If all win-lose games derived from a game form are determined (*via* strategies in R_1, R_2), the game form is also said to be determined (*via* strategies in R_1 and R_2).
- Let $P \subseteq O$, and let $s_1 \in S_1$ such that $v(s_1, S_2) := \{v(s_1, s_2) \mid s_2 \in S_2\} \subseteq P$. The strategy s_1 is said to enforce P and exclude $O \backslash P$.

Examples of derived games

	2_l	2_r
1_t	X	Y
1_b	Y	X

	2_l	2_r
1_t	X	Z
1_b	Y	Y

	2_l	2_m	2_r
1_t	X	Z	Y
1_b	Y	Y	Y

Lifting the preferences

The main theorem of this paper needs in the proof a lifting of preferences $\prec$ to sets, i.e., we must define what it means for an agent to prefer a set of outcomes over another set.

Lifting the preferences

The main theorem of this paper needs in the proof a lifting of preferences $\prec$ to sets, i.e., we must define what it means for an agent to prefer a set of outcomes over another set.
"Old" definition:

$$\forall A, B \subseteq S, \quad A \prec^{\mathcal{P}} B \ := \ \exists a \in A \backslash B, \forall b \in B \backslash A, \ a \prec b$$

Rest of the construction then required $\prec$ to be a strict linear order.

Lifting the preferences

The main theorem of this paper needs in the proof a lifting of preferences $\prec$ to sets, i.e., we must define what it means for an agent to prefer a set of outcomes over another set.
"Old" definition:

$$\forall A, B \subseteq S, \quad A \prec^{\mathcal{P}} B \;:=\; \exists a \in A \backslash B, \forall b \in B \backslash A,\, a \prec b$$

Rest of the construction then required $\prec$ to be a strict linear order.
Contribution of this work (on the paper-and-pencil front): Using an alternative lifting that does not require $\prec$ to be linear.

Finitary equilibrium transfer

Theorem

Let $\langle\{1,2\}, S_1, S_2, O, v, \{\prec_1, \prec_2\}\rangle$ *be a two-player game where* O *is finite and let us assume the following:*

1. *The game form is determined via strategies in* R_1 *and* R_2.
2. *Both preferences* $\prec_1$ *and* $\prec_2$ *are strict partial orders.*

Then the game $\langle\{1,2\}, S_1, S_2, O, v, \{\prec_1, \prec_2\}\rangle$ *has a Nash equilibrium in* $R_1 \times R_2$.

Finitary equilibrium transfer: Proof sketch

Theorem

Let $\langle\{1,2\}, S_1, S_2, O, v, \{\prec_1, \prec_2\}\rangle$ *be a two-player game where* O *is finite and let us assume the following:*

1. *The game form is determined via strategies in* R_1 *and* R_2*.*
2. *Both preferences* $\prec_1$ *and* $\prec_2$ *are strict partial orders.*

Then the game $\langle\{1,2\}, S_1, S_2, O, v, \{\prec_1, \prec_2\}\rangle$ *has a Nash equilibrium in* $R_1 \times R_2$*.*

Proof sketch:

1. Let M be the $\prec_1^{\mathcal{P}}$-greatest subset of O that Player 1 can enforce using strategy s_1.

Finitary equilibrium transfer: Proof sketch

Theorem

Let $\langle \{1,2\}, S_1, S_2, O, v, \{\prec_1, \prec_2\}\rangle$ be a two-player game where O is finite and let us assume the following:

1. *The game form is determined via strategies in R_1 and R_2.*
2. *Both preferences $\prec_1$ and $\prec_2$ are strict partial orders.*

Then the game $\langle \{1,2\}, S_1, S_2, O, v, \{\prec_1, \prec_2\}\rangle$ has a Nash equilibrium in $R_1 \times R_2$.

Proof sketch:

1. Let M be the $\prec_1^{\mathcal{P}}$-greatest subset of O that Player 1 can enforce using strategy s_1.
2. Let m be $\prec_2$-maximal in M, and let $M' := (M \setminus \{m\}) \cup u(m)$. One can see that $M \prec_1^{\mathcal{P}} M'$.

Finitary equilibrium transfer: Proof sketch

Theorem

Let $\langle\{1,2\}, S_1, S_2, O, v, \{\prec_1, \prec_2\}\rangle$ be a two-player game where O is finite and let us assume the following:

1. *The game form is determined via strategies in R_1 and R_2.*
2. *Both preferences $\prec_1$ and $\prec_2$ are strict partial orders.*

Then the game $\langle\{1,2\}, S_1, S_2, O, v, \{\prec_1, \prec_2\}\rangle$ has a Nash equilibrium in $R_1 \times R_2$.

Proof sketch:

1. Let M be the $\prec_1^{\mathcal{P}}$-greatest subset of O that Player 1 can enforce using strategy s_1.
2. Let m be $\prec_2$-maximal in M, and let $M' := (M \setminus \{m\}) \cup u(m)$. One can see that $M \prec_1^{\mathcal{P}} M'$.
3. Player 1 cannot enforce M'. So Player 2 can enforce $O \backslash M'$ using strategy s_1. It turns out that $v(s_1, s_2) = \{m\}$ and that (s_1, s_2) is a Nash equilibrium.

Some basics

- Standard Isabelle/HOL in ISAR proof style without any special libraries
- Restriction to two players!

Some basics

- Standard Isabelle/HOL in ISAR proof style without any special libraries
- Restriction to two players!
- Around 1100 lines of proof code.
- Many lines for technicalities concerning the lifting of $\prec$, e.g., showing that the lifted order is transitive (160 lines).

Games forms and games

There is nothing to define about strategies and the outputs: they are simply type parameters.

```
type_synonym ('O,'S1,'S2) game_form = "('S1 * 'S2) ⇒ 'O"

type_synonym ('O,'S1,'S2) game =
   "('O ⇒ 'O ⇒ bool) * ('O ⇒ 'O ⇒ bool) * (('O,'S1,'S2) game_form)"
```

Functions *pref1*, *pref2*, and *form* extract each of the three components of a game g.

Nash equilibrium and determined game

```
definition
  isNash :: "(('O,'S1,'S2) game) ⇒ 'S1 ⇒ 'S2 ⇒ bool"
  where "isNash g s1 s2 =
     ((∀ s1'. ¬(pref1 g) ((form g) (s1,s2)) ((form g) (s1',s2))) ∧
      (∀ s2'. ¬(pref2 g) ((form g) (s1,s2)) ((form g) (s1,s2'))))"

definition
  determined :: "((bool,'S1,'S2) game) ⇒ ('S1 set) ⇒ ('S2 set)
                                                          ⇒ bool"
  where "determined g R1 R2 =
                          ((∃ s1∈R1. ∀ s2. (form g) (s1,s2) = True) ∨
                           (∃ s2∈R2. ∀ s1. (form g) (s1,s2) = False))"
```

Derived win-lose game and determined game form

```
definition
  derivedWLGame :: "(('O,'S1,'S2) game_form) ⇒ ('O set) ⇒
((bool,'S1,'S2) game)"
  where "derivedWLGame gf Ou =
     ((λ ou p. p∧¬ou), (λ ou p. ou∧¬p), (λou. ou∈Ou)∘gf)"
```

Note the simplified outcome type!

```
definition
  determinedForm :: "(('O,'S1,'S2) game_form) ⇒ ('S1 set) ⇒ ('S2 set)
                                                              ⇒ bool"
  where "determinedForm gf R1 R2 =
                      (∀ Ou. determined (derivedWLGame gf Ou) R1 R2)"
```

Main result

```
theorem equilibrium_transfer_finite :
  assumes finite0 : "finite (range (form g))"
      and trans1 : "⋀a b c. (pref1 g) a b ⟹ (pref1 g) b c
                                              ⟹ (pref1 g) a c"
      and irref1 : "⋀a. ¬(pref1 g) a a"
      and trans2 : "⋀a b c. (pref2 g) a b ⟹ (pref2 g) b c
                                              ⟹ (pref2 g) a c"
      and irref2 : "⋀a. ¬(pref2 g) a a"
      and det : "determinedForm (form g) R1 R2"
  shows "∃s1∈R1. ∃s2∈R2. isNash g s1 s2"
```

153 lines of proof but uses various lemmas.

Overview of the formal setup in Coq

- Formalization choice: provide game-theoretic definitions (game form, Nash eq. . .) that are as general as possible before instantiating them to two-player games.

Overview of the formal setup in Coq

- Formalization choice: provide game-theoretic definitions (game form, Nash eq. . .) that are as general as possible before instantiating them to two-player games.
- The entire formalization has around 1300 lines of Coq code.
- 270 lines of Coq code are devoted to prove all properties of the lifting of $\prec$.

Overview of the formal setup in Coq

- Formalization choice: provide game-theoretic definitions (game form, Nash eq. . .) that are as general as possible before instantiating them to two-player games.
- The entire formalization has around 1300 lines of Coq code.
- 270 lines of Coq code are devoted to prove all properties of the lifting of $\prec$.
- Main dependency: SSReflect and MathComp
  - especially using theories `fintype`, `finfun`, `finset`, and `bigop`
  - $\rightsquigarrow$ comprehensive formalization of finite sets
  - $\rightsquigarrow$ facilities to reason about discrete objects in a “classical” way

Summary of the main definitions (1/3)

```
Variables (Agt : Type)(Strat : Agt → Type)(Outc : Type).
Definition strategy := ∀a : Agt, Strat a. (* dep. type *)
Record game_form := GameForm
{ preform :> strategy → Outc ;
  eq_strategy : (* extensionality property *) }.
Record game := Game
{ form :> game_form ;
  prefs : Agt → Outc → Outc → bool }.
Definition is_NE (g : game) (strat : strategy) : Prop :=
  ∀a : Agt,∀strat' : strategy,
  (∀b : Agt, a ≠ b → strat b = strat' b) →
  ¬ prefs g a (g strat) (g strat').
Definition ex_NE (g : game) : Type :=
  {strat : strategy | is_NE g strat}.
```

Summary of the main definitions (1/3)

```
Variables (Agt : Type)(Strat : Agt → Type)(Outc : Type).
Definition strategy := ∀a : Agt, Strat a. (* dep. type *)
Record game_form := GameForm
{ preform :> strategy → Outc ;
  eq_strategy : (* extensionality property *) }.
Record game := Game
{ form :> game_form ;
  prefs : Agt → Outc → Outc → bool }.
Definition is_NE (g : game) (strat : strategy) : Prop :=
  ∀a : Agt,∀strat' : strategy,
  (∀b : Agt, a ≠ b → strat b = strat' b) →
  ¬ prefs g a (g strat) (g strat').
Definition ex_NE (g : game) : Type :=
  {strat : strategy | is_NE g
```

read "∃"

Summary of the main definitions (2/3)

```
Inductive player := player1 | player2.
Definition game_form_2 := game_form player. (* instantiation *)
Definition game_2 := game player.
Inductive winlose_outc := win1 | win2.
Definition winlose_prefs (a: player) (o1 o2 : winlose_outc) :=
  match a, o1, o2 with
  | player1, win2, win1 ⇒ true
  | player2, win1, win2 ⇒ true
  | _, _, _ ⇒ false
  end.
Definition derivedWLGame :
  ∀Outc Strat, (Outc → winlose_outc) →
  game_form_2 Outc Strat → game_2 winlose_outc Strat.
```

Summary of the main definitions (2/3)

```
Inductive player := player1 | player2.
Definition game_form_2 := game_form player. (* instantiation *)
Definition game_2 := game player.
Inductive winlose_outc := win1 | win2.
Definition winlose_prefs (a: player) (o1 o2 : winlose_outc) :=
  match a, o1, o2 with
  | player1, win2, win1 ⇒ true
  | player2, win1, win2 ⇒ true
  | _, _, _ ⇒ false
  end.
Definition derivedWLGame :
  ∀Outc Strat, (Outc → winlose_outc) →
  game_form_2 Outc Strat → game_2 winlose_outc Strat.
```

player1 prefers win1

Summary of the main definitions (3/3)

```
Variables (Strat : player → Type)(Outc : Type).
Definition preferred_outc (a : player) : winlose_outc :=
  if a is player1 then win1 else win2.
Definition win_strat (v : game_form_2 winlose_outc Strat)
(a : player) (sa : Strat a) :=
  ∀s : strategy Strat, s a = sa → v s = preferred_outc a.
Definition determined (v : game_form_2 winlose_outc Strat) :=
  {a : player & {sa : Strat a | win_strat v a sa}}.
Definition determined_form (v : game_form_2 Outc Strat) :=
  ∀wl : Outc → winlose_outc, determined (derivedWLGame wl v).
```

The formalized theorem in a nutshell

```
Theorem finite_equilibrium_transfer :
 ∀(Strat : player → Type) (_ : strategy player Strat)
  (Outc : finType) (g : game_2 Outc Strat)
  (Strat_R : player → Type)
  (incl : ∀a : player, Strat_R a → Strat a),
 StrictOrder (prefs g player1) →
 StrictOrder (prefs g player2) →
 determined_form_via incl (form g) →
 ex_NE_via incl g.
```

The formalized theorem in a nutshell

```
Theorem finite_equilibrium_transfer :
 ∀(Strat : player → Type) (_ : strategy player Strat)
  (Outc : finType) (g : game_2 Outc Strat)
  (Strat_R : player → Type)
  (incl : ∀a : player, Strat_R a → Strat a),
 StrictOrder (prefs g player1) →
 StrictOrder (prefs g player2) →
 determined_form_via incl (form g) →
 ex_NE_via incl g.
```

- Focus on a **finite set of outcomes**
- Proved for **arbitrary strategy spaces**
- Axiom-free proof in Coq

Summary

- A dual formalization of a game-theoretic theorem in **Coq** and **Isabelle**.
- Involves key concepts such as **game forms** and **determinacy**.
- Mutual insemination between theory (paper-and-pencil proofs) & practice (formal proof) & between the 2 proof assistants.

+Isar classical logic eases the proofs

more readable scripts thanks to structured, declarative proofs

+Coq dependent types helpful to set general definitions

even if EM is not available, we can work in decidable fragments or make decidability hypotheses explicit

Perspectives

- feed our theorem (which transforms determinacy into $\exists$ of NE) with the positional determinacy of parity games $\rightsquigarrow$ Isabelle
- prove the full result by Le Roux (requires transfinite induction) $\rightsquigarrow$ easier in Coq than in Isabelle
- generalize the strict partial orders to acyclic binary relations $\rightsquigarrow$ doable in Coq and in Isabelle
- aim: provide a wider game theory formal library