Dependent Types for Extensive Games

Pierre Lescanne

Dependent Types for Extensive Games

Pierre Lescanne

University of Lyon, École normale supérieure de Lyon, CNRS (LIP),
46 allée d'Italie, 69364 Lyon, France

December 5, 2017

Abstract

Extensive games are tools largely used in economics to describe decision processes of a community of agents. In this paper we propose a formal presentation based on the proof assistant Coq which focuses mostly on infinite extensive games and their characteristics. Coq proposes a feature called "dependent types", which means that the type of an object may depend on the type of its components. For instance, the set of choices or the set of utilities of an agent may depend on the agent herself. Using dependent types, we describe formally a very general class of games and strategy profiles, which corresponds somewhat to what game theorists are used to. We also discuss the notions of infiniteness in game theory and how this can be precisely described.

Keywords: extensive game, infinite game, sequential game, coinduction, Coq, proof assistant.

1 Introduction

Extensive games are used in formalization of economics and in decision processes. Rational decision is logic, but it is not exaggerated to claim that rational decision is essentially a computational process and therefore it should be based on computational logic, like the calculus of inductive construction of Coq and on induction. Moreover, an adequate description of the decision process requires the framework to be infinite. Indeed there is no reason to assume that the process is a priori finite, since if we do so we put strong constraints on the model which prevents some behaviors, like for instance escalation. Beware, in the framework of games where agents interact, we do not say that the world is infinite, but we say that the agents believe that the world is infinite. Indeed, saying that the model is finite precludes the phenomenon of escalation, and proving, in that case, that escalation cannot exist is begging the question. Since we require a computational approach to infinite processes, the natural concept in modern logic is this of coinduction as proposed in [16, 13, 15]. But in this paper, by using dependent types, we revise our previous works. Thus we allow considering formal presentations of very general classes of games, for instance, games with very general sets of choices depending on agents or very general sets of utilities also depending on agents. For instance, an agent may have an infinity of choices and another may have only one choice, or two, whereas utilities are just ordered sets, even completely trivial ones in some counterexamples, which shows their generality. Similarly agents may have their own sets of utility. Agents may prefer flowers for their colors whereas agents use their fragrances. By very small changes in the formalism, we may easily describe multistage games, that are games in which agents move simultaneously at each stage.

All the formalism has been developed in `Coq` [2]. The reader can find scripts on GitHub at
`https://github.com/PierreLescanne/DependentTypesForExtensiveGames`.

The paper has 8 sections. The second section presents games and strategy profiles. Section 3, Section 4 and Section 6 talk about concepts connected with finiteness. Section 8 considers the way infiniteness is addressed in books on game theory. Section 9 is the conclusion.

2 Games and Strategy Profiles

This presentation of extensive games differs from this of [16, 13, 15, 1] in the use of dependant types. However it has connections with composition games [7, 11]. Indeed, for simplicity, in those papers, only binary games were considered[1], that is that only two choices were offered to the agents. In this paper, using dependent types, we can propose a more general framework. Associated with a game, a strategy profile is a description of the choices taken by the agents. The formal definitions of games and strategy profiles relies on three entities, a set of agents written `Agent`, a set of choices depending on an agent `a` written `Choice a` and a set of utilities depending on an agent `a` written `Utility a`. Moreover there is a preorder on `Utility a`. In particular, unlike most of the presentations of games, utilities need not be natural numbers, but can be any ordered set used by the agent. The sets of infinite games and of infinite strategy profiles are defined coinductively and are written respectively `Game` and `StratProf`.

Game.

A game which does not correspond to a terminal position and which we call a node is written `<|a,next|>` and has two arguments:

- an *agent* `a`, the agent whom the node belongs to,
- a function `next` of type `Choice a` $\to$ `Game`.

We call *leaf* a terminal position. A leaf consists in a function

[1] After Vestergaard [23] who introduced this concept for finite games and finite strategy profiles.

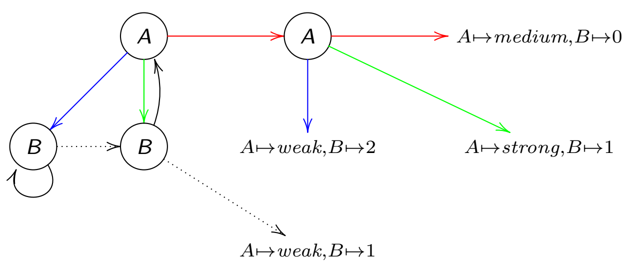
$$(\forall \texttt{ a:Agent, Utility a}) \to \texttt{Game}$$

i. e., a function form an agent `a` to and element of `Utility a`, which is the utility assignment at the end of the game and which is written `<| f |>`. Notice that the utility depends on the agent. A node game is made of an agent and of a function which returns a game given a choice. Assume that the agent is `a` and the function is `next`, then a node game is written `<|a,next|>`. The formal definition of a game is given in Coq by:

```
CoInductive Game : Set :=
| gLeaf :  (forall a:Agent, Utility a) -> Game
| gNode : forall (a:Agent), (Choice a -> Game) -> Game.
```

Since this defines a *coinductive*, this covers finite and infinite extensive games.

Example 1 *Here is game with* choices *blue, green and red for* A *and* black *and* dotted *for* B *and* $\{weak, medium, strong\}$ *as* utilities *for* A*, and* $\mathbb{N}$ *as* utilities *for* B*.*



Strategy profile.

A strategy profile corresponds to a non terminal position. We call it a node and we write it $\ll$`a,c,next`$\gg$. It has three components:

- an *agent* `a`, the agent whom the node belongs to,
- a *choice* `c`, which is the choice taken by agent on this specific node,
- a function `next` of type `Choice a` $\to$ `StratProf`.

A strategy profile which is a terminal position is a function

$$(\forall \texttt{ a:Agent, Utility a}) \to \texttt{Game}$$

like for games. Indeed there is no choice. It is written `<<f>>`. The inductive definition in Coq of a strategy profile is:

```
CoInductive StratProf  : Set :=
| sLeaf : (forall a:Agent, Utility a) -> StratProf
| sNode : forall (a:Agent),
            Choice a -> (Choice a -> StratProf) -> StratProf.
```

The two main differences with the approach of [16, 13, 15, 1] lie in the fact that the set of choices and the set of utilities are not fixed (the same for all agents, namely a pair) but depend on the agent (dependent type). This way we can describe a larger class of games. In Example 1, we have shown a game with choices and games actually depending on the agents. For instance, as we will see in Section 4, the sets of choices can easily be infinite. Since the built-in Coq equality is not adequate, we define coinductively an equality on games,

```
CoInductive gEqual: Game -> Game -> Prop :=
| gEqualLeaf: forall f, gEqual (<| f |>) (<| f |>)
| gEqualNode: forall (a:Agent)(next next':Choice a->Game),
  (forall (c:Choice a), gEqual (next c) (next' c)) ->
  gEqual (<|a,next|>) (<|a,next'|>).
```

further written `==`.

Utility assignment.

Since Coq accepts only terminating functions we define the utility assignment as a relation:

```
Inductive Uassign : StratProf ->  (forall a:Agent, Utility a) -> Prop :=
| UassignLeaf: forall f, Uassign (<<f>>) f
| UassignNode: forall  (a:Agent)(c:Choice a)
                       (ua: forall a',Utility a')
                       (next:Choice a -> StratProf),
    Uassign (next c) ua  -> Uassign (<<a,c,next>>) ua.
```

We prove that `Uassign` is a functional relation, namely that

```
 forall s ua ua', Uassign s ua -> Uassign s ua' -> ua=ua'..
```

Notice that for proving this property we need an inversion tactic which is somewhat subtle when dealing with dependent types [4, 17].[2] Moreover for all convergent strategy profiles (i.e., for all strategy profiles of interest, see next section) we can prove that the function is total, i.e., that there exists always a utility assignment associated with this convergent strategy profile.

[2]We thank Adam Chlipala and Jean-François Monin for their help on this specific example.

3 Several notions associated with finiteness

On infinite games and strategy profiles there are several predicates capturing notions of finiteness.

Finite Games.

A game is finite if it has a finite number of positions. It is naturally an inductive[3]. Clearly a leaf is finite. A game which is a node is finite if the set of the choices of the agent is finite [4] and if for all the choices, the next games are finite. This is made precise by the following definition.

```
Inductive Finite : Game -> Set :=
| finGLeaf: forall f, Finite <|f|>
| finGNode: forall (a:Agent)(next: Choice a -> Game),
              finite (Choice a) ->
              (forall c:Choice a, Finite (next c)) ->
              Finite <|a,next|>.
```

Finite strategy profiles would be defined likewise.

```
Inductive FiniteStratProf : StratProf -> Set :=
| finSLeaf: forall f, FiniteStratProf <<f>>
| finSNode: forall (a:Agent)(c:Choice a)(next: Choice a -> StratProf),
              finite (Choice a) ->
              (forall c':Choice a, FiniteStratProf (next c')) ->
              FiniteStratProf <<a,c,next>>.
```

Games with only finitely many strategy profiles.

Osborne and Rubinstein [19] call “finite”, a game with only finitely many strategy profiles[5]. In order not to interfere with the previous definition, we prefer to say that *the game is finitely broad.*[6] This is translated by the fact that for a game `g` to have only finitely many strategy profiles, there shall exist a list that collects all the strategy profiles that have this game `g` as underlying game. Since in Coq lists are finite this yields the desired property:

```
Definition FinitelyBroad (g:Game): Prop :=
  exists (l: list StratProf), forall (s:StratProf),
      game s == g <-> In s l.
```

[3]Roughly speaking, an inductive (definition) is a well-founded definition with basic cases and constructors

[4]The predicate `finite` over choices is not defined here.

[5]Actually they use the concept of “history” (path), instead of strategy profiles, but this is not essential.

[6]Denoted by the predicate `FinitelyBroad` on `Game` in Coq

Games with only finite histories.

A game has only finite histories if it has only finitely many paths (histories) from the root to the leaves. This can be described as follows:

```
Inductive FiniteHistoryGame : Game -> Prop :=
| finHorGLeaf: forall f, FiniteHistoryGame <|f|>
| finHorGNode: forall (a:Agent)(next: Choice a -> Game),
             (forall c':Choice a, FiniteHistoryGame (next c')) ->
             FiniteHistoryGame <|a,next|>.
```

Those games should not be confused with games with finite horizon. Notice that Osborne and Rubinstein [19] require a game with a finite horizon to have only finitely many strategy profiles (p. 90: "[Given a finite game] if the longest history is finite then the game has finite horizon"), whereas Osborne [18] does not require the set of strategy profiles associated to the game to be finite (see Section 8). For strategy profiles we have:

```
Inductive FiniteHistoryStratProf : StratProf -> Prop :=
| finHorSLeaf: forall f, FiniteHistoryStratProf <<f>>
| finHorSNode: forall (a:Agent) (c:Choice a)
                      (next: Choice a -> StratProf),
             (forall c':Choice a, FiniteHistoryStratProf (next c')) ->
             FiniteHistoryStratProf <<a,c,next>>.
```

Convergent strategy profiles.

The finiteness does not apply to all paths (histories) leading to leaves, but applies only to paths corresponding to the choices of the agents. Mutatis mutandi, the expression

```
(forall c':Choice a, FiniteHistoryStratProf (next c'))
```

is just replaced by

```
Convergent (next c)
```

hence without the

```
forall c':Choice a
```

Related to induction reasoning, this convergence of strategy profiles captures continuity. Like for the predicate `FiniteHistoryGame` a leaf is convergent. A strategy profile which is a node is convergent if the strategy subprofile for the choice made by the agent a (i.e., next c) is convergent.

```
Inductive Convergent: StratProf -> Prop :=
| ConvLeaf: forall f, Convergent <<f>>
| ConvNode: forall (a:Agent) (c:Choice a)
                   (next: Choice a -> StratProf),
             Convergent (next c) ->
             Convergent <<a,c,next>>.
```

The reader may notice the similarity of that definition with this of finite histories for games. We are now able to prove a theorem on the totality of Uassign:

```
Lemma ExistenceUassign:
  forall (s:StratProf),
    (Convergent s) -> exists (ua: forall a, Utility a), Uassign s ua.
```

Convergence is extended to all the strategy subprofiles of a given strategy profile by a modality Always, abbreviated □, when used in expressions. Always applies to a predicate on StratProf i.e. a function P:StratProf → Prop Always P s means that P is fulfilled by all subprofiles of s.

```
CoInductive Always (P:StratProf -> Prop) : StratProf -> Prop :=
| AlwaysLeaf : forall f, Always P (<<f>>)
| AlwaysNode : forall (a:Agent)(c:Choice a)
                      (next:Choice a->StratProf),
          P (<<a,c,next>>) ->  (forall c', Always P (next c')) ->
               Always P (<<a,c,next>>).
```

The predicate Always Convergent is shortened in ⇓. ⇓ s means that s is convergent and also all subprofiles are convergent. It plays a main role in the definition of other concepts related to strategy profiles, namely equilibria and escalation. Always convergent strategy profiles are the right objects, that game theorists are interested in. "Always Convergence" captures the notion of continuity in the spirit of Brouwer [3].[7]

4 A game with only finite histories and no longest history

In this section we show how Coq can be used to prove formally properties about games. Specifically we give an example of a game with only finite histories and no longest history as a counterexample to Osborne (see [18] p. 157) definition of finite horizon. The game has two agents whom we call Alice and Bob and its definition uses a feature of dependent types, namely that the choices may

[7] I like to thank Jules Hedges for pointing me this fact and the connection with Brouwer bar recursion [10].

depend on the agent. In this case, `Alice` has infinitely many choices, namely the set `nat` of natural numbers and `Bob` has one choice, namely the set `unit`. The *utility* of *Alice* and *Bob* are meaningless since they are singletons, namely the `Coq` built-in `unit` which contains the only element `tt`. In `Coq` we have:

```
Definition Choice :(AliceBob -> Set) :=
  fun a:AliceBob  => match a with Alice => nat | Bob => unit end.
```

and

```
Definition Utility: AliceBob -> Set := fun a => unit.
```

Notice that `Choice` and `Utility` are functions which take an agent and return a set. Said otherwise, the set of choices is the result of the function `Choice` applied to agents and the set of utilities is the result of the function `Utility` applied to agents. If the agent is `Alice`, the set of choices is `nat` and the set of utility is `unit`. If the agent is `Bob` the set of choices and the set of utilities are `unit` (a singleton). In other words, the set of choices depends on the agents and the set of utilities looks depending on the agents, but doesn't. The game has infinitely many threadlike subgames of length n:

```
Fixpoint  ThreadlikeGame (n:nat): (Game AliceBob Choice Utility) :=
  match n with
    | 0 => <|fun (a:AliceBob) => match a with | Alice => tt
                                              | Bob => tt end|>
    | (S n) => <|Bob,fun c:Choice Bob
                      => match c with tt=>ThreadlikeGame n end|>
  end.
```

The game we are interested in is called `GameWFH` and is defined as a node with agent `Alice` and with next games `ThreadlikeGame n` for Alice's choice `n`:

```
Definition GameWFH:(Game AliceBob Choice Utility)  :=
  <| Alice, fun n:Choice Alice => ThreadlikeGame n |>.
```

Let us call `triv` the utility assignment `Alice => tt, Bob => tt`. We can picture `GameWFH` like in Figure 1. One can prove that `ThreadlikeGame n` has only finite histories:

```
Proposition FiniteHistoryGameWFH:
    FiniteHistoryGame AliceBob Choice Utility GameWFH.
```

Clearly `GameWFH` has no longest history.

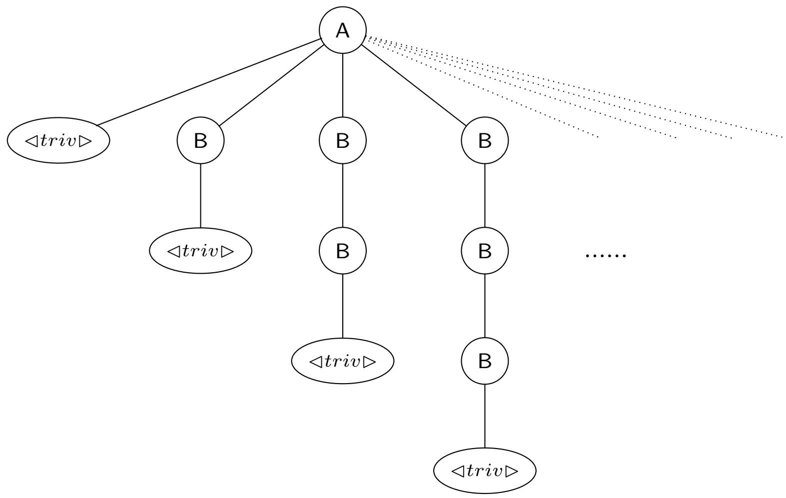


Figure 1: Picture of game with finite histories and no longest history

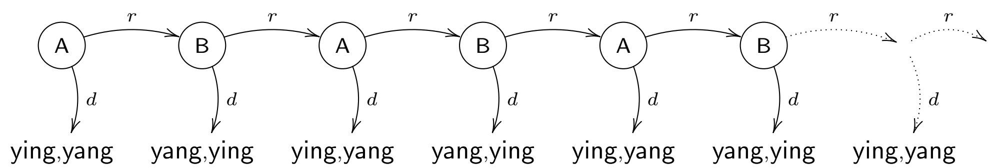
5 Subgame Perfect Equilibrium

An agent is rational if her strategy is based on a strategy profile which is a subgame perfect equilibrium. So let us present *subgame perfect equilibria*. Subgame perfect equilibria are specific strategy profiles that fulfill some "good" properties. Therefore they are presented by a predicate which we call `SPE`. In Coq this is a function of type `StratProf -> Prop`. A strategy profile, which is a node, is a subgame perfect equilibrium if first it is always convergent. This is necessary to be able to compute the utility assignment. Moreover the choice of the agent is better than or equal to other choices w.r.t. to the utility assignment and all the strategy subprofiles of this strategy profile are themselves subgame perfect equilibria. A leaf is a subgame perfect equilibrium. This can be formalized in Coq:

```
CoInductive SPE : StratProf -> Prop :=
| SPELeaf : forall (f: forall a:Agent, Utility a), SPE <<f>>
| SPENode :  forall (a:Agent)
                    (c c':Choice a)
                    (next:Choice a->StratProf)
                    (ua ua':forall a':Agent, Utility a'),
               Always convergent <<a,c,next>> ->
               Uassign (next c') ua' ->  Uassign (next c) ua ->
               (pref a (ua' a) (ua a)) -> SPE (next c') ->
             SPE <<a,c,next>>.
```

6 The simplest escalation

We discussed already the rationality of escalation in infinite games [16, 15]. For dependent choice games, escalation is a somewhat simple concept and consists in adjusting the types. The simplest escalation is probably as follows. It may occur in a game in which there are two agents `Alice` and `Bob`, where each agent has two choices `down` and `right` and in which there are two non ordered utilities `ying` and `yang`. We use `ying` and `yang` to insist on the fact that there is no need for numbers and no need for an actual order among the utility values.



This is basically the game studied in [15], with the difference that the preference in `Utility = {ying, yang}` is just the equality. In other words, agents do not need to prefer one item over the other, just a trivial preference may lead to an escalation. The agents are like Buridan's ass [24], they may not know what to choose and therefore go forever. This may look strange, but as shown by the Coq script, the proof is based on exactly the same proof technique as this of the rationality of the escalation of the dollar auction [22] as shown by the two following Coq statements and proofs:[8]

```
Lemma AlongGoodAndDivergentInDollar :
  exists (s:StratProf dollar.Agent dollar.Choice dollar.Utility),
    AlongGood dollar.Agent dollar.Choice dollar.Utility dollar.pref s
    /\ Divergent s.
Proof.
  exists (dollarAcBc 0).
  split.
  apply AlongGoodDolAcBc.
  apply DivergenceDolAcBc.
Qed.
```

and the proof of the escalation for the *YingYang game*:

```
Lemma AlongGoodAndDivergentInYingYang :
  exists (s:StratProf yingYang.Agent yingYang.Choice yingYang.Utility),
    AlongGood yingYang.Agent yingYang.Choice yingYang.Utility yingYang.pref s
    /\ Divergent s.
Proof.
  exists yingYangAcBc.
  split.
  apply AlongGoodYyAcBc.
  apply DivergenceYyAcBc.
Qed.
```

7 Multi-stage games

Multi-stage games are introduced in [6] (Section 3.2). We view them as games in which a node does not belong to an agent and the choices or the moves of all the agents are simultaneous. Let us call `MSGame` the multi-stage games. The simultaneous or collective choice corresponds to the type:

```
(forall a:  Agent, Choice a) -> MSGame
```

or written with products:

$$\prod_{\mathtt{a}\in\mathtt{Agent}} \mathtt{Choice\ a}.$$

[8] Notice that the parameters of StratProf are explicit!

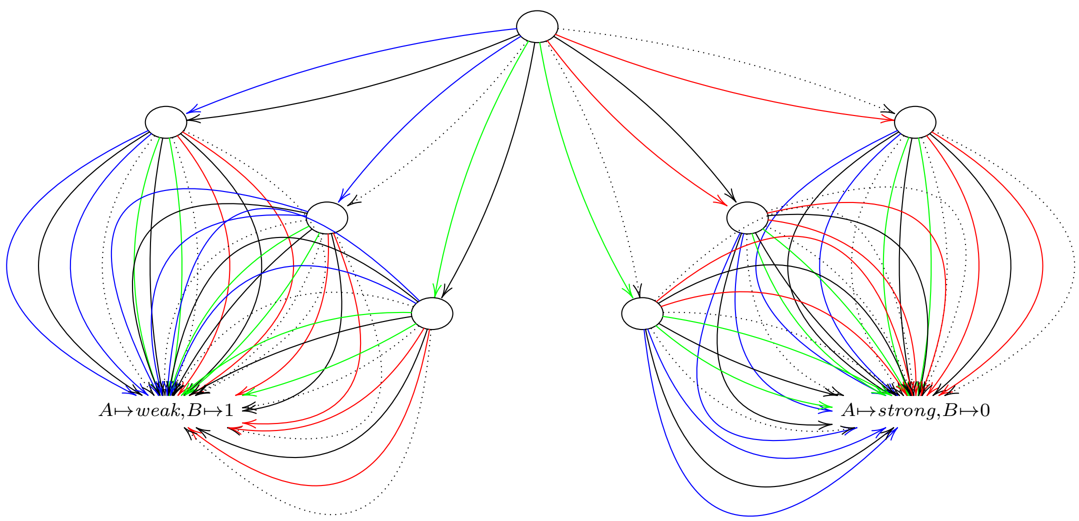
Leaves are almost unchanged. The function `next` is of type

$$\texttt{next:} \quad (\prod_{\texttt{a}\in\texttt{Agent}} \texttt{Choice a}) \to \texttt{MSGame}$$

and a node is just the function next:

```
CoInductive MSGame :=
| msgLeaf: (forall a: Agent, Utility a) -> MSGame
| msgNode: ((forall a: Agent, Choice a) -> MSGame) -> MSGame.
```

Example 2 *T show the complexity of multistage games, we draw a picture of a simple multistage game with the same choices and utilities as Example 1.*



8 Infinite and infinite

In this section, we look at the way infiniteness is dealt with in textbooks on game theory.

Two views of infiniteness

Infiniteness is discussed by Poincaré in his book *Science et méthode* [20], where he distinguishes *mathematical infinite* which we would call today *potential infinite*, and *actual infinite*. Poincaré did not believe in such an actual infinite, but today we do accept a concept of actual infinite which is the foundation of the theory of coinduction and infinite games. Let us discuss these two concepts in the case of words on the alphabet $\{a, b\}$. $\{a, b\}^+$ represents all the (finite) words made with the letters a and b, like $a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb$, *etc.* One can also write:

$$\{a, b\}^+ \quad = \quad \bigcup_{n=0}^{\infty} \{a, b\}\,\{a, b\}^n. \qquad (1)$$

$\{a, b\}^+$ is the least fixpoint of the equation:

$$X = \{a, b\} \cup \{a, b\}X$$

There are infinitely many such words. This is a first kind of infinite, indeed we can build words of all finite lengths. $\{a, b\}^\omega$ is the set of infinite words. Each infinite word can be seen as a function $\mathbb{N} \to \{a, b\}$. An infinite word represents another kind of infinite. For instance the infinite word *ababab...* or $(ab)^\omega$ corresponds to the function $if\ \mathsf{even}(n)\ then\ a\ else\ b$ and is a typical example of actual infinite. $\{a, b\}^\omega$ is solution of the fixpoint equation:

$$X = \{a, b\}X.$$

In $\{a, b\}^+$ there is no infinite objects, but only approximations, whereas in $\{a, b\}^\omega$ there are only infinite objects.

Figure 2 represents the two notions of infiniteness. On the left, the vault ceiling of Nasir ol Molk Mosque in Chiraz[9] pictures potential infiniteness. On the right, a drawing[10] inspired by M.C. Escher Waterfall pictures actual infiniteness.

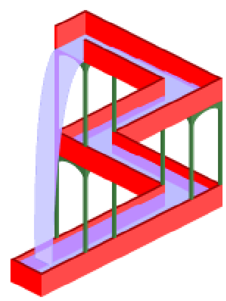


Figure 2: Two pictures of infinite

Common Knowledge.

Common Knowledge is a central concept in game theory, it relies on the concept of knowledge of an agent, which is a modality i.e., an operator of modal logic.[11] Modality K_a (knowledge of agent a) follows the laws of modal logic S_5. For this and the group G of agents, we create a modality E_G (shared knowledge):

$$E_G(\varphi) = \bigwedge_{a \in G} K_a(\varphi).$$

The common knowledge modality is

$$C_G(\varphi) = \bigwedge_{n=0}^{\infty} E_G^n(\varphi). \quad (2)$$

[9]From Wikimedia, due to User:Pentocelo.

[10]From Wikimedia.

[11]We follows the presentation of [12, 14], which took its origin from [5].

Usually there is no ambiguity on the group of agents, thus instead C_G and E_G one write just C and E. Clearly C has the flavor of $+$ as shown by the analogy between equation (1) and equation (2) and their fixpoint definitions.

Infinite and fixpoint.

Infinite objects are associated with fixpoints. For instance, $\{a, b\}^+$ is the least fixpoint of the equation:

$$X \quad = \quad \{a, b\} \cup \{a, b\}X$$

whereas $C(\varphi)$ is the least fixpoint of

$$X \quad \Leftrightarrow \quad \varphi \wedge E(X).$$

which means that $C(\varphi)$ is a solution:

$$C(\varphi) \quad \Leftrightarrow \quad \varphi \wedge E(C(\varphi)).$$

Infinite in textbooks.

In general in textbooks on game theory “infinite” is a vague notion which in not defined precisely and words like “ad infinitum” ([6] p. 542, [9] p. 27) or “infinite regress” ([6] p. 543) or three dots are used. It is often said that infinite games resemble repeated games, but this is not true, since repeated games are typically potential infinite presentations of infinite games, i.e., approximation – only sequences of games are considered, not their limit – whereas infinite games are defined by coinduction.

Two main mistakes are worth noticing.

- In [9], Hargreaves and Varoufakis define common knowledge as follows:

  (a) each person is instrumentally rational

  (b) each person knows (a)

  (c) each person knows (b)

  (d) each person knows (c)

  . . . and son on *ad infinitum*.

  but they add “The idea reminds one of what happens when a camera is pointing to a television screen that conveys the image recorded by the very same camera : *an infinite self-reflection*”, showing that they clearly mixed up the two kinds of notions. Indeed clearly the infinite self-reflection illustrates an actual infinite, a little like the infinite word $(ab)^\omega$ or the Escher waterfall, whereas, as we said, common knowledge is a potential infinite. An expression like *ad libitum* should have been preferred and the image of a swing going further and further or a tessellation, like this of Figure 2 should have been more appropriate.

- In [18], Osborne uses the “length of longest terminal history” to define finite horizon, without checking whether this longest history actually exists. A counterexample is shown in Section 4. We gather that he means the “least upper bound on $\overline{\mathbb{N}}$ of the lengths of the histories”.

9 Conclusion

If, when reaching this point, the reader has the feeling that there is no proof or almost no proof, this means that she (he) did not read the Coq files of the GitHub site, as indicated in the introduction. In those files, there is nothing but proofs. But those proofs which are mostly meant to be read by a computer are, at the present time, not part of a scientific paper [8].

The formalization of infinite extensive games in Coq is only at an early stage. Among possible tracks to develop, there is the connection between multistage games and extensive (one-stage) games, that is between games where players move simultaneously and games where players play in alternation, using moves “do nothing” (see [6] p. 70). More precisely we do not know how to intepret the sentence of Fudenberg and Tirole:

> *Common usage to the contrary “simultaneous moves” does not exclude games where players move in alternation, as we allow for the possibility that some of the players have the one-element choice set “do nothing”.*

References

[1] Samson Abramsky and Viktor Winschel. Coalgebraic analysis of subgame-perfect equilibria in infinite games without discounting. *arXiv*, 2012.

[2] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, Gérard Huet, Henri Laulhère, César Muñoz, Chetan Murthy, Catherine Parent-Vigouroux, Patrick Loiseleur, Christine Paulin-Mohring, Amokrane Saïbi, and Benjamin Werner. *The Coq Proof Assistant Reference Manual*. INRIA, version 6.3.11 edition, May 2000.

[3] John L. Bell. Continuity and infinitesimals. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2014 edition, 2014.

[4] Adam Chlipala. *Certified Programming with Dependent Types*. MIT Press, 2011. `http://adam.chlipala.net/cpdt/`.

[5] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[6] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.

[7] Neil Ghani and Jules Hedges. A compositional approach to economic game theory. *CoRR*, abs/1603.04641, 2016.

[8] Thomas C. Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *CoRR*, abs/1501.02155, 2015.

[9] Shaun Hargreaves-Heap and Yanis Varoufakis. *Game Theory: A Critical Introduction*. Routledge, 2004.

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

[11] Jules Hedges, Paulo Oliva, Evguenia Shprits, Viktor Winschel, and Philipp Zahn. Selection equilibria of higher-order games. In Yuliya Lierler and Walid Taha, editors, *Practical Aspects of Declarative Languages - 19th International Symposium, PADL 2017, Paris, France, January 16-17, 2017, Proceedings*, volume 10137 of *Lecture Notes in Computer Science*, pages 136–151. Springer, 2017.

[12] Pierre Lescanne. Mechanizing common knowledge logic using COQ. *Ann. Math. Artif. Intell.*, 48(1-2):15–43, 2006.

[13] Pierre Lescanne. Rationality and escalation in infinite extensive games. *CoRR*, abs/1112.1185, 2011.

[14] Pierre Lescanne. Common knowledge logic in a higher order proof assistant. In Andrei Voronkov and Christoph Weidenbach, editors, *Programming Logics - Essays in Memory of Harald Ganzinger*, volume 7797 of *Lecture Notes in Computer Science*, pages 271–284. Springer, 2013.

[15] Pierre Lescanne. A simple case of rationality of escalation. In Reiko Heckel and Stefan Milius, editors, *CALCO*, volume 8089 of *Lecture Notes in Computer Science*, pages 191–204. Springer, 2013.

[16] Pierre Lescanne and Matthieu Perrinel. "Backward" coinduction, Nash equilibrium and the rationality of escalation. *Acta Inf.*, 49(3):117–137, 2012.

[17] Jean-François Monin and Xiaomu Shi. Handcrafted inversions made operational on operational semantics. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 2013.

[18] Martin J. Osborne. *An Introduction to Game Theory*. Oxford, 2004.

[19] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, Massachusetts, 1994.

[20] Henri Poincaré. *Science et méthode*. Ernest Flammarion, Paris, 1908. Translated in English as [21].

[21] Henri Poincaré. *Science and Method*. Thomas Nelson and Sons, 1914. Translated by Francis Maitland, available online at `https://archive.org/details/sciencemethod00poinuoft`.

[22] Martin Shubik. The dollar auction game: A paradox in noncooperative behavior and escalation. *Journal of Conflict Resolution*, 15(1):109–111, 1971.

[23] René Vestergaard. A constructive approach to sequential Nash equilibria. *Inf. Process. Lett.*, 97:46–51, 2006.

[24] Wikipedia. Buridan's ass — wikipedia, the free encyclopedia, 2016. [Online; accessed 13-July-2016].