HAL open science

Deconstruction of Infinite Extensive Games using coinduction

Pierre Lescanne

To cite this version:

Pierre Lescanne. Deconstruction of Infinite Extensive Games using coinduction. 2009. ensl-00376141v2

HAL Id: ensl-00376141

https://ens-lyon.hal.science/ensl-00376141v2

Preprint submitted on 28 Apr 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Deconstruction of Infinite Extensive Games using Coinduction

Pierre Lescanne*

Université de Lyon, ENS de Lyon, CNRS (LIP),
46 allée d'Italie, 69364 Lyon, France

April 28, 2009 – 15 : 13

Abstract. Finite objects and more specifically finite games are formalized using induction, whereas infinite objects are formalized using *coinduction*. In this article, after an introduction to the concept of coinduction, we revisit on infinite (discrete) extensive games the basic notions of game theory. Among others, we introduce a definition of *Nash equilibrium* and a notion of *subgame perfect equilibrium* for infinite games. We use those concepts to analyze well known infinite games, like the *dollar auction game* and the *centipede game* and we show that human behaviors that are often considered as illogic are perfectly rational, if one admits that human agents reason coinductively.

1 Introduction

Finite extensive games have been introduced by Harold Kuhn [17]. But many interesting extensive games are infinite and therefore the theory of infinite extensive games play an important role in game theory, with examples like the *dollar auction game* [35,6,15,29] or the generalized *centipede game*.[1] But, from a formal point of view, they are not appropriately treated in papers and books. In particular, there is no clear notion of Nash equilibrium and the gap between finiteness and infiniteness is not correctly understood. Instances of vague definitions related to infiniteness are in [30] p. 157 and in [29] p. 90, where the authors speak about the *length of the longest history* without guaranteeing that such a longest history exists. More precisely, Osborne in [30] writes: *If the length of the longest derivation is [...] finite, we say that the game has a finite horizon. Even a game with a finite horizon may have infinitely many terminal histories, because some player has infinitively many actions after some history.* If a game has only finite histories, but has infinitely many such finite histories of increasing length, the length of the longest history does not exist and it is not clear whether such a game should be considered to have a *finite horizon* or not.

Another common mistake arises in escalation [35]. It is to believe that results on infinite games can be obtained as the limit of results on finite games. It is notorious that there is a threshold (a wall) between finiteness and infiniteness,

* This research has been supported by Région *Ile de France*.

[1] Here *"generalized"* means that the game has an infinite "backbone".

a fact known to model theorists [13,12] and to specialists of real variable functions, as shown by the Weierstrass function, a counterexample of the fact that a finite sum of functions differentiable everywhere is differentiable everywhere whereas an infinite sum can be differentiable nowhere. In this kind of inadequate reasoning, people study finite games that are the infinite games truncated at a finite bound, then they extrapolate their results to infinite games by increasing the size to infinity. But this says nothing since the *infiniteness is not the limit of finiteness*. They use this to conclude that humans are wrong in their reasoning. But clearly, if there would be a limit, then there would be no escalation. In other words, if there is an escalation, then the game is infinite, then *the reasoning must be specific to infinite games, i.e., based on coinduction* and this is only on this basis that one can conclude that humans are rational or irrational and for us it turns out that agents are rational in escalating. In short, *Macbeth is rational.*

In this paper we address these issues and the games we consider may have arbitrary long histories as well as infinite histories. In our games there are two choices at each node, this is without loss of generality, since we can simulate finitely branching games in this framework. By König's lemma, finitely branching, specifically binary, infinite games have at least an infinite history. We are taking the problem of defining formally infinite games, infinite strategy profiles, and infinite histories extremely seriously. Another important issue which is not considered in the literature is how the utilities associated with an infinite history are computed. To be formal and rigorous, we expect some kinds of recursive definitions, more precisely co-recursive definitions, but then comes the questions of what the payoff associated with an infinite strategy is and whether such a payoff exists.

Finite extensive games are represented by finite trees and are analyzed through induction. For instance, in finite extensive games, a concept like *subgame perfect equilibrium* is defined inductively and receives appropriately the name of *backward induction*. Similarly *convertibility* (an agent changes choices in his strategy) has also an inductive definition and this concept is a key for this of *Nash equilibrium* which is not itself given by an inductive definition. But *induction*, which has been designed for finitely based objects, no more works on infinite[2] games, i.e., games represented by infinite trees. Logicians have proposed a tool, which they call *coinduction*, to reason on infinite objects. In short, since objects are infinite and their construction cannot be analyzed, coinduction "observes" them, that is looks at how they "react" to operations (see Section 2 for more explanation). In this paper, we formalize with coinduction, the concept of infinite game, of infinite strategy profile, of equilibrium in infinite games, of utility (payoff), and of subgame and we verify on the proofs assistant Coq that everything works smoothly and yields interesting consequences. Thanks to coinduction, some examples of apparently paradoxical human behavior are explained logically, demonstrating a rational behavior.

[2] In this paper, *infinite* means infinite and discrete. For us, an infinite extensive game is discrete and has infinitely many nodes.

Induction vs coinduction. To formalize structured finite objects, like finite games, one uses *induction*, i.e., a definition of basic objects (in the case of finite games they are leafs or terminal nodes) and a definition of the way to build new objects (in the case of finite games induction provides an operation to build a game from subgames). In the case of infinite objects like infinite games, this no more works and one characterizes infinite objects by their behavior. But this raises two questions: what happens if I query the utility (payoff) of an infinite strategy profile? What happens if I ask for a subgame of an infinite game? This characterization by "observation" is called *coinduction*. Whereas induction is associated with the least fixed point of a definition, coinduction is associated with the greatest fixed point. The proof assistant Coq offers a framework for coinductive definitions and reasonings which are keys of our formalization.

Structure of the paper. Deconstruction is a sketch of what coinduction does, but we can say that in this paper we undertake a deconstruction of the concept of infinite games using coinduction, which was made possible by the extreme rigor and discipline imposed by the proof assistant Coq. The proof assistant Coq was indeed crucial for the author, but this paper tries to hide this aspect.

The paper is structured as follows. First we present the notion of infinite object using this of *history* (Section 2), we use this example to introduce the concept of coinduction. Then we explore the notions of *infinite game* (Section 3) and *infinite strategy* (the word we use for *strategy profile*) (Section 4). The concept of *convertibility* (Section 6) allows us to introduce the concepts of *equilibrium* (Section 7). We apply those concepts to two infamous examples (Section 8), namely the *dollar auction game* and the *centipede game* and we say a few words about human reasoning versus coinductive reasoning (Section 8.3). In Section 9 we tell how *coinduction* is implemented in Coq and how it is used.[3] Related works are presented in Section 10. Appendix gives formal version of the definitions that are given in the paper, they are also for the reader interested by the formal aspects.

Coq scripts. The documentation of the Coq scripts containing only statements of definitions and lemmas is given in

`http://perso.ens-lyon.fr/pierre.lescanne/COQ/INFGAMES/`

It contains what has been proved without the proofs themselves. The Coq scripts, i.e., the detail of the proofs, are in

`http://perso.ens-lyon.fr/pierre.lescanne/COQ/INFGAMES/SCRIPTS/`

This part is meant for those with some expertise in Coq and who are interested by the correction of the logical arguments.

[3] This section can be dropped by readers less interested by the behavior of Coq.

2 An example of infinite objects: histories

Infinite objects have peculiar behaviors. To start with a simple example, let us have a look at *histories* in games. In a game, agents make *choices*. In an infinite game, agents can make finitely many choices before ending, if they reach a terminal node, or infinitely many choices, if they run forever. Choices are recorded in a *history* in both cases. A history is therefore a finite or an infinite list of choices. In this paper, we consider that there are two possible choices: ℓ and r (ℓ for "left" and r for "right"). Since a history is a potentially infinite object, it cannot be defined by structural induction.[4] On the contrary, the type[5] *History* has to be defined as a Coinductive, i.e., by coinduction. Let us use the symbol $[\,]$ for the empty history and the binary operator :: for non empty histories. When we write $c :: h$ we mean the history that starts with c and follows with the history h. To define histories coinductively we say the following:

A **coinductive** history (or a finite or infinite history) is
- either the empty history $[\,]$,
- or a history of the form $c :: h$, where c is a choice and h is a history.

The word "coinductive" says that we are talking about finite or infinite objects. This should not be mixed up with finite histories which will be defined inductively as follows:

An **inductive** history (or a finite history) is built as
- either the empty history $[\,]$,
- or a finite non empty history which is the composition of a choice c with a finite history ℓ to make the finite history $c :: \ell$.

Notice the use of the participial "built", since in the case of induction, we say how objects are built, because they are built finitely. Let us now consider four families of histories:

$\mathcal{H}_0$	The family of finite histories
$\mathcal{H}_1$	The family of finite histories or of histories which ends with an infinite sequence of ℓ's
$\mathcal{H}_2$	The family of finite histories or infinite histories which contains infinitely many ℓ's
$\mathcal{H}_\infty$	The family of finite or infinite histories

We notice that $\mathcal{H}_0 \subset \mathcal{H}_1 \subset \mathcal{H}_2 \subset \mathcal{H}_\infty$. If $\mathcal{H}$ is a set of histories, we write $c :: \mathcal{H}$ the set $\{h \in \mathcal{H}_\infty \mid \exists h' \in \mathcal{H}, h = c :: h'\}$. We notice that $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ are solutions of the fixpoint equation :

$$\mathcal{H} = \{[\,]\} \ \cup \ \ell :: \mathcal{H} \ \cup \ \mathsf{r} :: \mathcal{H}.$$

[4] In type theory, a type of objects defined by induction is called an Inductive, a shorthand for *inductive type*.

[5] Since we are in type theory, the basic concept is this of *"type"*. Since we are using only a small part of type theory, it would not hurt to assimilate naive types with naive sets.

in other words

$$\begin{aligned}
\mathcal{H}_0 &= \{[\,]\} \;\cup\; \ell :: \mathcal{H}_0 \;\cup\; \mathsf{r} :: \mathcal{H}_0 \\
\mathcal{H}_1 &= \{[\,]\} \;\cup\; \ell :: \mathcal{H}_1 \;\cup\; \mathsf{r} :: \mathcal{H}_1 \\
\mathcal{H}_2 &= \{[\,]\} \;\cup\; \ell :: \mathcal{H}_2 \;\cup\; \mathsf{r} :: \mathcal{H}_2 \\
\mathcal{H}_\infty &= \{[\,]\} \;\cup\; \ell :: \mathcal{H}_\infty \;\cup\; \mathsf{r} :: \mathcal{H}_\infty
\end{aligned}$$

Among all the fixpoints of the above equation, $\mathcal{H}_0$ is the least fixpoint and describes the inductive type associated with this equation, that is the type of the finite histories and $\mathcal{H}_\infty$ is the greatest fixpoint and describes the coinductive type associated with this equation, that is the type of the infinite and infinite histories. The principle that says that given an equation, the least fixpoint is the inductive type associated with this equation and the greatest fixpoint is the coinductive type associated with this equation is very general and will be used all along this paper.

In the Coq vernacular, we are more talkative, but also more precise to describe the CoInductive type *History*, (see appendix B for the definition.). Recall the definition

> A **coinductive** history (or a finite or infinite history) is
> – either the empty history [ ],
> – or a history of the form $c :: h$, where c is a choice and h is a history.

which says that a *history* is either the null history or a history which is a history, finite or infinite, increased by a choice at its head. The word **coinductive** guarantees that we define actually infinite objects and attach to the objects of type *History* a specific form of reasoning, called *coinduction*. In coinduction, we assume that we "know" an infinite object by observing it through its definition, which is done by a kind of peeling. Since on infinite objects there is no concept of being smaller, one does not reason by saying "I know that the property holds on smaller objects let us prove it on the object". On the contrary one says "Let us prove a property on an infinite object, for that peel the object, assume that the property holds on the peeled object and prove that it holds on the whole object". One does not say that the object is smaller, just that the property holds on the peeled object. The above presentation is completely informal, but it has been, formally founded in the theory of Coq, after the pioneer works of David Park [32] and Robin Milner [24], using the concept of greatest fixpoint in type theory [7]. Bertot and Castéran [4] present the concepts in Chapter 13 of their book.

By just observing them, one cannot prove that two objects which have exactly the same behavior are equal, we can just say that they are observably equivalent. Observable equivalence is a relation weaker than equality[6], called *bisimilarity* and defined on *History* as a CoInductive (see appendix for a fully formal definition in the Coq vernacular):

[6] We are talking here about *Leibniz equality*, not about *extensional equality* see appendix A.

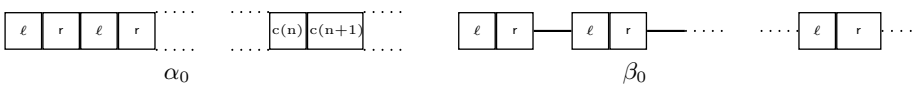


Fig. 1. The picture of two bisimilar histories

Bisimilarity $\sim_h$ on *histories* is defined **coinductively** as follows:
- $[\,] \sim_h [\,]$,
- $h \sim_h h'$ implies $\forall a : Agent, a :: h \sim_h a :: h'$.

This means that two histories (why not infinite) are bisimilar if both are null or for composed histories, if both have the same head and the rests of both histories are bisimilar. One can prove that two objects that are equal are bisimilar, but not the other way around, because for two objects to be equivalent by observation, does not mean that they have the same structure. To illustrate the difference between bisimilarity and equality of infinite objects let us consider for example two infinite histories α_0 and β_0 that are obtained as solutions of two equations, more precisely as two cofixpoints. Let $\alpha_n = c(n) :: \alpha_{n+1}$, where $c(n)$ is (**if** $even(n)$ **then** ℓ **else** r), and β_p is $\ell :: \mathsf{r} :: \beta_{p+1}$. We know that if we ask for the 5th element of α_0 and β_0 we will get ℓ in both cases, and the $2p^{th}$ element will be r in both cases, but we have no way to prove that α_0 and β_0 are equal, i.e., have exactly the same structure. Actually the picture in Figure 1 shows that they look different and there is no hope to prove by induction, for instance, that they are the same, since they are not well-founded.

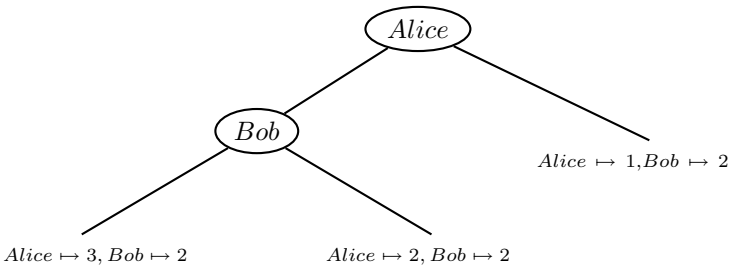
3 Infinite Games

In classical textbooks, finite and infinite games are presented through their histories. But in the framework of a proof assistant or just to make rigorous proofs, it makes sense to present them structurally. Therefore, games are rather naturally seen as either a leaf to which a *utility function* (a function that assigns a utility to each agent, aka an *outcome*) is attached or a node which is associated to an agent and two subgames. If agents are *Alice* and *Bob* and utilities are natural numbers, a utility function can be the function $\{Alice \mapsto 3, Bob \mapsto 2\}$. In Section 8.2, we focus on *centipede games* which are games in which the right part is always a leaf and in which the left subgame is always a composed game. Actually we study games that "can" be infinite and "can" have finite or infinite branches.

The type of *Game*s is defined as a **coinductive** as follows:
- a *Utility function* makes a *Game*,
- an *Agent* and two *Game*s make a *Game*.

A *Game* is either a leaf (a terminal node) or a composed game made of an agent (the agent who has the turn) and two subgames (the formal definition in

the Coq vernacular is given in the appendix B). We use the expression *gLeaf f* to denote the leaf game associated with the utility function f and the expression *gNode a* g_l g_r to denote the game with agent a at the root and two subgames g_l and g_r. For instance, the game we would draw:



is represented by the term:
$(gNode\ Alice\ (gLeaf\ {\scriptstyle Alice \mapsto 1, Bob \mapsto 2}\ )$
$(gNode\ Bob\ (gLeaf\ {\scriptstyle Alice \mapsto 2, Bob \mapsto 2})\ (gLeaf\ {\scriptstyle Alice \mapsto 3, Bob \mapsto 2}))$

Hence one builds a finite game in two ways: either a given utility function f is encapsulated by the operator *gLeaf* to make the game (*gLeaf f*), or an agent a and two games g_l and $g_r$are given to make the game (*gNode a* g_l g_r). Notice that in such games, it can be the case that the same agent a has the turn twice in a row, like in the game (*gNode a* (*gNode a* g_1 g_2) g_3).

Concerning comparisons of utilities we consider a very general setting where a utility is no more that a type (a "set") with a preference which is a preorder, i. e., a transitive and reflexive relation, and which we write $\preceq$. A preorder is enough for what we want to prove. We assign to the leaves, a utility function which associates a utility to each agent.

We can also tell how we associate a history with a game or a history and a utility function with a game (see the Coq script). We will see in the next section how to associate a utility with an agent in a game, this is done in the frame of a strategy, which is described now.

4 Infinite Strategies

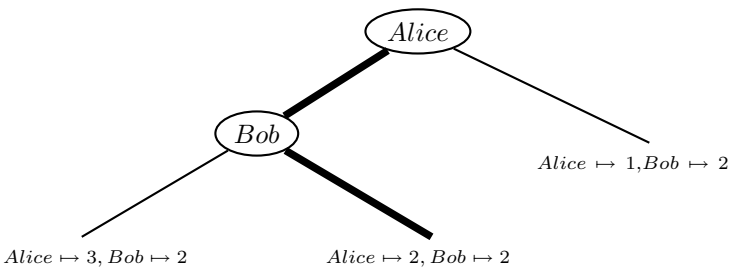
The main concept of this paper is this of infinite strategy[7] which is a coinductive. More specifically, in this paper, we focus on infinite binary strategies associated with infinite binary games.

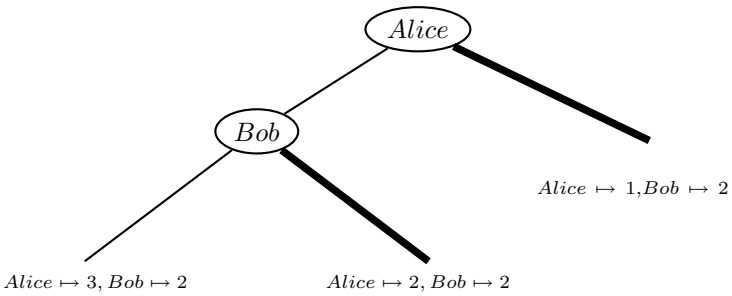
The type of *Strategies* is defined as a **coinductive** as follows:
– a *Utility function* makes a *Strategy*.

[7] Authors call it a *strategy profile*, but in this paper, following Vestergaard [38], we call it a *strategy* for convenience in the formal development and we give it a formal definition.

– an *Agent*, a *Choice* and two *Strategies* make a *Strategy*.

Basically[8] an infinite strategy which is not a leaf is a node with four items: an agent, a choice, two infinite strategies. A strategy is the same as a game, except that there is a choice. In what follows, since we consider equilibria, we only address strategies. Strategies of the first kind are written $\ll f \gg$ and strategies of the second kind are written $\ll a, c, s_l, s_r \gg$. In other words, if between the "$\ll$" and the "$\gg$" there is one component, this component is a utility function and the result is a leaf strategy and if there are four components, this is a node strategy. For instance, with the game of page 7 one can associate at least the following strategies:





which correspond to the expressions

$$\ll Alice, \mathsf{r}, \ll Alice \mapsto 1, Bob \mapsto 2 \gg, \ll Bob, \ell, \ll Alice \mapsto 2, Bob \mapsto 2 \gg, \ll Alice \mapsto 3, Bob \mapsto 2 \gg \gg$$

and

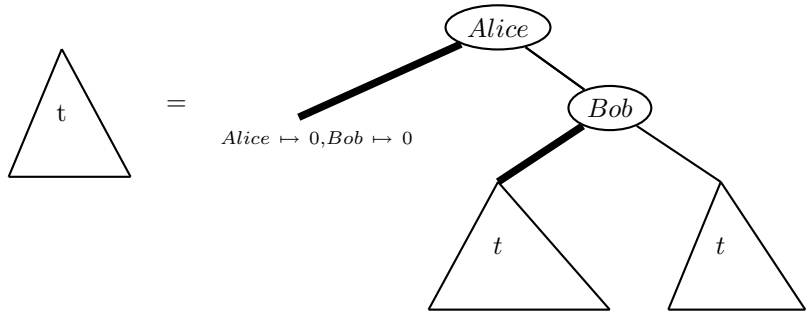
$$\ll Alice, \ell, \ll Alice \mapsto 1, Bob \mapsto 2 \gg, \ll Bob, \ell, \ll Alice \mapsto 3, Bob \mapsto 2 \gg, \ll Alice \mapsto 2, Bob \mapsto 2 \gg, \gg.$$

To describe an infinite strategy one uses most of the time a fixpoint equation like:

$$t \quad = \quad \ll Alice, \ell, \ll Alice \mapsto 0, Bob \mapsto 0 \gg, \ll Bob, \ell, t, t \gg \gg$$

[8] The formal definition in the Coq vernacular is given in appendix B.

which correspond to the pictures:



Other examples of infinite strategies are given in Section 8. Usually an infinite game is defined as a cofixpoint, i.e., as the solution of an equation, possibly a parametric equation.

Whereas in the finite case one can easily associate with a strategy a utility function, i.e., a function which assigns a utility to an agent, as the result of a recursive evaluation, this is no more the case with infinite strategies. One reason is that one is not sure that such a utility function exists for the strategy. This makes the function partial, which cannot be defined as inductive or coinductive. Therefore $s2u$ (an abbreviation for *strategy-to-utility*) is a relation between a strategy and a utility function, which is also a coinductive; $s2u$ appears in expression of the form $(s2u\ s\ a\ u)$ where s is a strategy, a is an agent and u is a utility. It reads "u is a utility of the agent a in the strategy s".

$s2u$ is a predicate defined **coinductively** as follows:
- $s2u \ll f \gg\ a\ (f(a))$ holds,
- if $s2u\ s_l\ a\ u$ holds then $s2u\ \ll a', \ell, s_l, s_r \gg\ a\ u$ holds,
- if $s2u\ s_r\ a\ u$ holds then $s2u\ \ll a', \mathsf{r}, s_l, s_r \gg\ a\ u$ holds.

This means the utility of a for the leaf strategy $\ll f \gg$ is $f(a)$, i.e., the value delivered by the function f when applied to a. The utility of a for the strategy $\ll a', \ell, s_l, s_r \gg$ is u if the utility of a for the strategy s_l is u. If one call s_0 the first above strategy, one has $s2u\ s_0$ *Alice* 2, which means that, for the strategy s_0, the utility of *Alice* is 2. In order to insure that $s2u$ has a result we define an operator *"leads to a leaf"* that says that if one follows the choices shown by the strategy one reaches a leaf, i.e., one does not go forever.

The predicate *"leads to a leaf"* is defined **inductively** as
- the strategy $\ll f \gg$ *"leads to a leaf"*,
- if s_l *"leads to a leaf"*, then $\ll a, \mathsf{l}, s_l, s_r \gg$ *"leads to a leaf"*,
- if s_r *"leads to a leaf"*, then $\ll a, \mathsf{r}, s_l, s_r \gg$ *"leads to a leaf"*.

This means that a strategy which is itself a leaf *"leads to a leaf"* and if the strategy is a node, if the choice is ℓ and if the left substrategy *"leads to a leaf"* then the whole strategy *"leads to a leaf"* and similarly if the choice is r. We

claim that this gives a good notion of *finite horizon* which seems to be rather a concept on strategies than on games.

If s is a strategy that satisfies the predicate *"leads to a leaf"* then the utility exists and is unique, in other words:

- For all agent a and for all strategy s, if s *"leads to a leaf"* then there exists a utility u which "is a utility of the agent a in the strategy s".
- For all agent a and for all strategy s, if s *"leads to a leaf"*, if "u is a utility of the agent a in the strategy s" and "v is a utility of the agent a in the strategy s" then $u = v$.

We also consider a predicate *"always leads to a leaf"* which means that everywhere in the strategy, if one follows the choices, one leads to a leaf. This property is defined everywhere on an infinite strategy and is therefore coinductive.

The predicate *"always leads to a leaf"* is defined **coinductively**
- the strategy $\ll f \gg$ *"always leads to a leaf"*,
- for all choice c, if $\ll a, \mathsf{r}, s_l, s_r \gg$ *"leads to a leaf"*, if s_l *"always leads to a leaf"*, if s_r *"always leads to a leaf"*, then $\ll a, \mathsf{r}, s_l, s_r \gg$ *"always leads to a leaf"*.

This says that a strategy, which is a leaf, *"always leads to a leaf"* and that a composed strategy inherits the predicate from its substrategies provided itself *"leads to a leaf"*. We define also bisimilarity between games and between strategies. For strategies, this is defined by:

The bisimilarity $\sim_s$ on *strategies* is defined **coinductively** as follows:
- $\ll f \gg \sim_s \ll f \gg$,
- if $s_l \sim_s s'_l$ and $s_r \sim_s s'_r$ then $\ll a, c, s_{,l}\, s_r \gg \sim_s \ll a, c, s'_l, s'_r \gg$.

This says that two leaves are bisimilar if and only if they have the same utility function and that two strategies are bisimilar if and only if they have the same head agent, the same choice and bisimilar substrategies.

5 Histories of strategies and finite strategies

In this short section, we present concepts that are not directly connected with equilibria, but that are of interest. For a finite or infinite history to be one of the histories of a game is characterized by a predicate. We define also the history of a strategy which is the history that consists in following the choices and which is typically obtained by a function. This function is a function on infinite objects since it associates an infinite object (a history) with an infinite object (a strategy). This kind of functions are defined as cofixpoints and are specific to infinite objects. Without surprise, the history of a strategy "is a history of" the game associated with that strategy. For a strategy to be finite is characterized by a unary predicate. If the history of a strategy is infinite then the strategy itself is infinite.

6 Convertibility

To speak about equilibria, one needs to speak first about a relation between strategies which is called *convertibility*. In classical game theory, a strategy s is given and a convertible strategy (r_a, s_{-a}) is the strategy in which agent a chooses r_a while other agents b choose s_b. We call the relation that relates s with (r_a, s_{-a}) *convertibility* and we write it $\vdash^a \dashv$. Basically two convertible strategies have the same "skeleton", i.e., the same underlying structure.

The relation $\vdash^a \dashv$ is defined **inductively** as follows:

- $\vdash^a \dashv$ is reflexive, i.e., for all s, $s \vdash^a \dashv s$.
- If the node has the same agent as the agent in $\vdash^a \dashv$ then the choice may change, i.e.,

$$\frac{s_1 \vdash^a \dashv s_1' \qquad s_2 \vdash^a \dashv s_2'}{\ll a, c, s_1, s_2 \gg \ \vdash^a \dashv \ \ll a, c', s_1', s_2' \gg}$$

- If the node does not have the same agent as in $\vdash^a \dashv$, then the choice has to be the same:

$$\frac{s_1 \vdash^a \dashv s_1' \qquad s_2 \vdash^a \dashv s_2'}{\ll a', c, s_1, s_2 \gg \ \vdash^a \dashv \ \ll a', c, s_1', s_2' \gg}$$

Since the relation is defined inductively it compares only a finite part of the strategies. Moreover convertibility preserves the predicate *AlwLeadsToLeaf*. This is illustrated by the following lemma:

For all a, s, s', if s *"always leads to a leaf"* and $s \vdash^a \dashv s''$ then s' *"always leads to a leaf"*.

Actually there is another convertibility which we write $\twoheadleftarrow^a \twoheadrightarrow$:

The relation $\twoheadleftarrow^a \twoheadrightarrow$ is defined **coinductively** as follows:

- A leaf is only convertible to itself.
- For nodes, one has basically the rules:

$$\frac{s_1 \twoheadleftarrow^a \twoheadrightarrow s_1' \qquad s_2 \twoheadleftarrow^a \twoheadrightarrow s_2'}{\ll a, c, s_1, s_2 \gg \ \twoheadleftarrow^a \twoheadrightarrow \ \ll a, c', s_1', s_2' \gg}$$

$$\frac{s_1 \twoheadleftarrow^a \twoheadrightarrow s_1' \qquad s_2 \twoheadleftarrow^a \twoheadrightarrow s_2'}{\ll a', c, s_1, s_2 \gg \ \twoheadleftarrow^a \twoheadrightarrow \ \ll a', c, s_1', s_2' \gg}$$

From their definition, one can prove that both convertibilities are equivalence relations. The difference between the two convertibilities appears only in the Coq script and shows typically the difference between induction and coinduction. It can be sketched as follows: $\twoheadleftarrow^a \twoheadrightarrow$ traverses the whole strategies and can accept infinitely many changes in choices, whereas $\vdash^a \dashv$ inspects only finite parts of the strategies, i.e., finitely many choices. We claim that human agents apply the convertibility $\vdash^a \dashv$ as they can only carry a finite amount of reasoning and comparison. Notice that when two strategies are convertible there underlying games are bisimilar.

7 Equilibria

7.1 Nash equilibria

The notion of Nash equilibrium is translated from the notion in textbooks. Since we have two concepts of convertibility, we might have two concepts of Nash equilibria, but as said we focus only on the inductive notion of convertibility $\vdash^a\dashv$. The concept of Nash equilibrium is based on a comparison of utilities; this assumes that an actual utility exists and therefore this requires convertible strategies to *"lead to a leaf"*. s is a *Nash equilibrium* if the following implication holds: if s *"leads to a leaf"* and for all agent a and for all strategy s' which is convertible to s, i.e., $s \vdash^a\dashv s'$, and which *"leads to a leaf"*, if u is the utility of s for a and u' is the utility of s' for a, then $u' \preceq u$.

7.2 Subgame Perfect Equilibria

Let us consider now *subgame perfect equilibria*, which we write $SGPE$. $SGPE$ is a property of strategies. It requires the substrategies to fulfill coinductively the same property, namely to be a $SGPE$, and to insure that the strategy with the best utility for the node agent to be chosen. Since both the strategy and its substrategies are infinite, it makes sense to make the definition of $SGPE$ coinductive.

$SGPE$ is defined **coinductively** as follows:
- $SGPE \ll f \gg$,
- if $\ll a, \ell, s_l, s_r \gg$ *"always leads to a leaf"*, if $SGPE(s_l)$ and $SGPE(s_r)$, if $s2u\ s_l\ a\ u$ and $s2u\ s_r\ a\ v$, if $v \leq u$ then $SGPE\ \ll a, \ell, s_l, s_r \gg$,
- if $\ll a, \mathsf{r}, s_l, s_r \gg$ *"always leads to a leaf"*, if $SGPE(s_l)$ and $SGPE(s_r)$, if $s2u\ s_l\ a\ u$ and $s2u\ s_r\ a\ v$, if $u \leq v$ then $SGPE\ \ll a, \mathsf{r}, s_l, s_r \gg$,

It means that a strategy, which is a leaf, is a subgame perfect equilibrium (condition *SGPE_Leaf*). Moreover if the strategy is a node, if the strategy *"always leads to a leaf"*, if it has agent a and choice ℓ, if both substrategies are subgame perfect equilibria and if the utility of the agent a for the right substrategy is less than this for the left substrategy then the whole strategy is a subgame perfect equilibrium and vice versa (condition *SGPE_left*). If the choice is r (condition *SGPE_right*) this works similarly.

Notice that since we require that the utility can be computed not only for the strategy, but for the substrategies and for the subsubstrategies and so on, we require these strategies no only to *"lead to a leaf"* but to *"always lead to a leaf"*.

8 Two infinite games with centipede shape

In this section we study two kinds of games that have some analogies, especially they have a centipede shape, since they have an infinite backbone (on the "left")

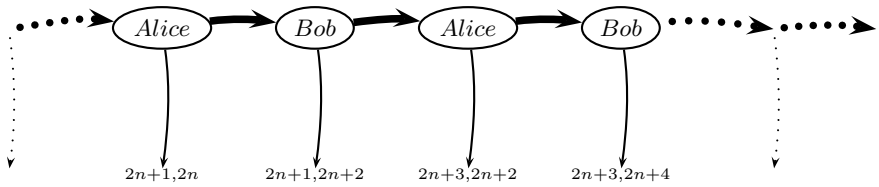


Fig. 2. The strategy "never give up" for the *dollar auction game* (payoffs are $100v$ minus the numbers)

and all the right subgames are leaves. In both cases, the utilities go to infinity, but in the first (dollar auction game) the utilities go to $(-\infty, -\infty)$ (costs, i.e., the opposites of utilities, go to $(+\infty, +\infty)$), whereas in the second, (centipede game) the utilities go to $(+\infty, +\infty)$. One common mistake in analyzing those games is to believe that results on the infinite game can be obtain as the limit of partial results on the finite games, which is not the case (see [13] for the perspective of a model theorist).

8.1 The "Illogic of Conflict Escalation"

The above development has been applied to an example called the *dollar auction game* by Shubik [35] and the *illogic conflict of escalation* by Gintis [15]. Recall its principle. Two agents *Alice* and *Bob* compete in an auction for an object of a value v \$ ($100v$ ¢). The two agents bid 2 ¢, one after the other. We assume that there is no limit on the bids, i.e., the bankroll of the agents is not limited. This generalization is possible because we reason on infinite games. By the way, if there would be a limit on the bids, there would be no escalation. If one agent gives up, the highest bidder gets the object, but the second bidder pays also for his (her) bid. As Shubik noted, this game may never cease, due to what Colman [6] calls the *Macbeth effect*. We have closely studied only two infinite strategies: a strategy called *always give up* (*agu* in short) (see Figure 2) and a strategy called *never give up* (*ngu* in short) (see Figure 3). In the strategy *agu*, both agents leave the auction at each step, when in the strategy *ngu*, the agents keep bidding, this is the escalation. *agu* and *ngu* are solutions of parametric equations. For instance *agu* is solution of the equation (cofixpoint):

$$agu_n \ = \ll Alice, \ell, \ll Bob, \ell, agu_{n+1}, [2n+1, 2n+2] \gg, [2n+1, 2n] \gg .$$

where $[x, y]$ stands for $\ll Alice \mapsto x, Bob \mapsto y \gg$. We are able to prove in Coq that *agu* is a *subgame perfect equilibria* and that both *agu* and *ngu* are Nash equilibria. The proof of *agu* being a subgame perfect equilibrium is based on a coinductive argument, whereas the proof of *agu* being a Nash equilibrium is by case; *ngu* is a Nash equilibrium because it does not "lead to a leaf", this last fact comes from a coinductive argument.

8.2 Centipede

The centipede games have been introduced by [33] (see also [5,29]), but only as finite games. The paradox is that the subgame perfect equilibrium is when the player do not enter the game, whereas clearly if they would enter the game, they would get a better payoff. In this development, we consider an infinite game and show that the strategy that never stops is also a Nash equilibrium; in other words if players enter the game they should not stop. Payoffs are given in Figure 4. Like for the *dollar auction game*, we consider two strategies, namely a strategy *always give up* and a strategy *never give up* and we can prove by coinduction that the first is both a subgame perfect equilibrium and a Nash equilibrium and that the second is a Nash equilibrium. The proofs are very similar to this for the *dollar auction game*. The strategy *never give up* is a Nash equilibrium, for the same reason: the agent cannot compute the payoff, but there is a difference, whereas the payoff cannot be computed in the *dollar auction game* because it goes to $-\infty$, in the case of the *centipede game* it cannot be computed because it goes to $+\infty$.

8.3 Human and formal reasoning

Literature (see [6] Section 9.3 for a detailed review) cites many experiments which have been used to compare human reasoning and formal reasoning and in the so called *illogic conflict of escalation* like in the *centipede game*, it is advocated that humans are illogic (see for instance [28,19]). Our work shows that this is not the case as we are able to prove that the infinite escalation is also a Nash equilibrium.[9] In other words when an agent decides to enter the game, then he (she) has no other rational solution but to bid forever (*Macbeth effect*) and acting so makes perfect sense from the theoretical point of view of coinduction[10], despite when it applies to war, it is scaring. So the advice should

[9] We have not been able to prove formally in Coq that *always give up* and *never give up* are the only Nash equilibria, but this seems very likely.

[10] We propose to call this the *Jourdain effect*, from the character of *Molière*'s play the *Bourgeois genthilhomme* (The middle-class gentleman), delighted to learn that he

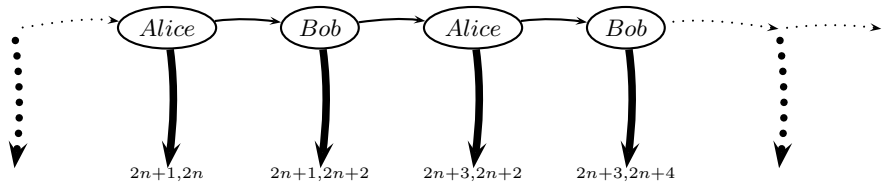


Fig. 3. The strategy "always give up" for the *dollar auction game* (payoffs are $100v$ minus the numbers)

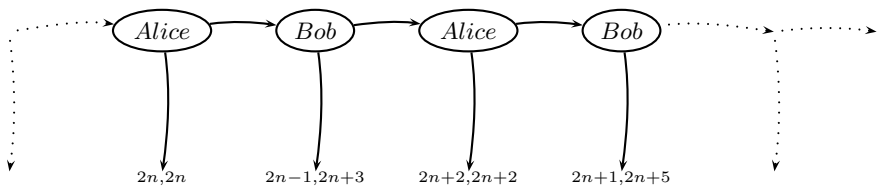


Fig. 4. The centipede game

be: "Never enter such a game, otherwise logic will push you to escalation". In games involving money, he (she) will then face bankruptcy, but this is another story. In both games, assuming that the agents commonly believe that the game is infinite, they have two possibilities, either not entering the game or entering it and running forever as both strategies are Nash equilibria.

9 Coinduction in Coq

As we have said the proof assistant Coq [22] plaid a central role in this research.

Why should we formalize a concept in a proof assistant? To answer this question we like to cite Donald Knuth [36]:

> *People have said you don't understand something until you've taught it in a class. The truth is you don't really understand something until you've taught it to a computer, until you've been able to program it.*

We claim that we can appropriately replace the last sentence by *"until you've taught it to a proof assistant, until you've code it into* Coq *[22], Isabelle [26], or PVS [31]"* as it seems even more demanding to "teach" a proof assistant like Coq than to write a program on the same topics. Actually without Coq, we would not have been able to capture the concepts of Nash equilibrium and Subgame Perfect equilibria presented in this paper. This is indeed the result of formal deduction, intuition and try and error in Coq since proving properties of infinite games and infinite strategy profiles is extremely subtle (see Section 9). Moreover by relying on a proof assistant, we can free this article from formal developments and tedious and detailed proofs, knowing anyway that they are correct in any detail and that the reader will refer to the Coq script in case of doubt. Therefore, we can focus on informal explanation. However, Coq proposes a readable, rigorous, and computer checked syntax, the *vernacular*, for

has been speaking prose all his life without knowing it. *"For more than forty years I have been speaking prose without knowing anything about it."*(act II, scene 5) [25].

definitions, lemmas and theorems and when we provide definitions in this paper, they are associated with expressions stated in the vernacular provide in the appendix. The vernacular should be seen as a XXIst century version of Leibniz' *characterica universalis* or Frege's *Begriffsshrift* [14].

Decomposing an object The principle of coinduction is based on the greatest fixpoint of the definition, that is a *coinduction defines a greatest fixpoint* (see [4]). There are two challenges when one works with such a principle: the difficulty of decomposing infinite objects and the invocation of coinduction. They are both presented in detail in [4], but let us describe them in few words. For the first problem, suppose one has a strategy s, which is not a leaf; one knows that s is of the form $\ll a, c, s_l, s_r \gg$ for some agent a, some choice c and some strategies s_l and s_r. To obtain such a presentation, one uses a mechanism which consists in defining a function identity on strategy which is a "clone" of $fun\ s \Rightarrow s\ end$:

Definition *Strategy_identity* (*s*:*Strategy*): *Strategy* :=
match s with
— $\ll f \gg \Rightarrow \ll f \gg$
— $\ll a,c,sl,sr \gg \Rightarrow \ll a,c,sl,sr \gg$
end.

In other words, the *strategy identity* function is, computationally speaking, the function which associates $\ll f \gg$ with $\ll f \gg$ and $\ll a,c,sl,sr \gg$ with $\ll a,c,sl,sr \gg$ and not the function which associates s with s. We can prove the *strategy decomposition* lemma:

Lemma *Strategy_decomposition*: $\forall$ *s*: *Strategy*,
Strategy_identity s = *s*.

Thus when one wants to decompose a strategy s, one replaces s by *Strategy_identity s* and one simplifies the expression, and one gets $\ll a, c, s_l, s_r \gg$ for some a, c, s_l and s_r.

Invoking coinduction The *principle of coinduction* is based on a *tactic*[11] called *cofix*. It consists in assuming the proposition one wants to proof, provided one applies it only on strict sub-objects. In the current implementation of Coq, the user has to ensure that he invokes it on "strict" sub-objects. This is not always completely trivial and requires a good methodology. However the *proof checker* (a piece of software which accepts only correct proofs) verifies that this constraint is fulfilled at the time of checking the proof.

10 Related works

This work started after this of Vestergaard [38] on finite games and finite strategies. We first developed proofs on finite strategies, but unlike Vestergaard who

[11] A *tactic* is a tool in Coq used to build proofs without using the most elementary constructions.

based his formalization on fixpoint definitions of predicates, we used only inductive definitions of predicates. Like Vestergaard, we were able to prove the main lemma of finite extensive games, namely that *backward induction strategy profiles are Nash equilibria*; the script is available at `http://perso.ens-lyon.fr/pierre.lescanne/COQ/INFGAMES/SCRIPTS/finite_games.v`. Overall, this "induction based" presentation allowed us to switch more easily to coinduction on infinite games. Beside this, a development in Coq of finite games with an arbitrary number of choices at any node has been made by Le Roux [34] (p. 83 and following) and an exploration of common knowledge, induction and Aumann's theorem on rationality has been proposed by Vestergaard, Ono and the author [39]. In [20], there is a presentation of a somewhat connected development in Coq, namely this of the *logic of common knowledge*.

Since we are talking about some computational aspects of games, people may make some analogies with other works, let us state what extensive games are not.

- Extensive games are not *semantic games* as presented in [1,37,21,16].
- Extensive games are not *logical games* used in proving properties of automata and protocols [23,2].
- This work has only loose connection with *algorithmic game theory* [27,10], which is more interested by the complexity of the algorithms, especially those which compute equilibria, than by their correction, and does not deal with infinite games.

Three published examples of Coq developments with coinduction are this on *real numbers* by Bertot [3], this on *temporal logic* by Coupet-Grimal [8] and this on *hardware verification* by Coupet-Grimal and Jakubiec [9]. The book [11] by Dowek gives a philosophical perspective of using a proof assistant based on type theory in mathematics.

Acknowledgments

I would like to thanks many persons for advice in the course of this research, among them I am especially indebted to Rene Vestergaard, Stephane Le Roux, Franck Delaplace, Herbert Gintis, Ralph Matthes, Philippe Audebaud, Olivier Laurent and Alexandre Miquel and last but not least Yves Bertot and Pierre Castéran for their beautiful *Coq'Art*.

11 Conclusion

Thanks to coinduction, we have reconciled human reasoning with rational reasoning in infinite extensive games. In other words, we claim that human agents reason actually by coinduction when faced to infinite games and are rational. Moreover we have shown once more the threshold between finiteness and infiniteness and that reasoning on infinite objects is not the limit when the size goes to infinity of reasoning on finite objects.

References

1. S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59:543–574, 1994.
2. Reynald Affeldt, Miki Tanaka, and Nicolas Marti. Formal proof of provable security by game-playing in a proof assistant. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec*, volume 4784 of *Lecture Notes in Computer Science*, pages 151–168. Springer, 2007.
3. Yves Bertot. Affine functions and series with co-inductive real numbers. *Mathematical Structures in Computer Science*, 17(nn):37–63, 2007.
4. Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development Coq'Art: The Calculus of Inductive Constructions*. Springer-Verlag, 2004.
5. A. M. Colman. *Rational Models of Cognition*, chapter Rationality assumptions of game theory and the backward induction paradox, pages 353–371. Oxford U. Press, 1998. Edited by Mike Oaksford and Nick Chater.
6. Andrew M. Colman. *Game theory and its applications in the social and biological sciences*. London New York : Routledge, 1999. Second edition.
7. Thierry Coquand. Infinite objects in type theory. In Henk Barendregt and Tobias Nipkow, editors, *TYPES*, volume 806 of *Lecture Notes in Computer Science*, pages 62–78. Springer, 1993.
8. Solange Coupet-Grimal. An axiomatization of linear temporal logic in the calculus of inductive constructions. *J Logic Computation*, 13(6):801–813, 2003.
9. Solange Coupet-Grimal and Line Jakubiec. Certifying circuits in type theory. *Formal Asp. Comput.*, 16(4):352–373, 2004.
10. Contantinos Daskalakis, Paul W. Golberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *Communications of the ACM*, 52(2):89–97, 2009.
11. G. Dowek. *Les métamorphoses du calcul*. Le Pommier, 2007. Grand Prix of Philosophy of the French Academy.
12. H. D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, New York, 1995.
13. R. Fagin. Finite model theory – a personal perspective. *Theoretical Computer Science*, 116:3–31, 1993.
14. Gottlob Frege. *From Frege to Gdel*, chapter Concept Script. Harvard Uni. Press., 1967.
15. Herbert Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2000.
16. Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical. Structures in Comp. Sci.*, 11(3):301–506, 2001.
17. Harold W. Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games II*, 1953. Reprinted in [18].
18. Harold W. Kuhn, editor. *Classics in Game Theory*. Princeton Uni. Press, 1997.
19. W. Leininger. Escalation and coooperation in conflict situations. *J. of Conflict Resolution*, 33:231–254, 1989.
20. P. Lescanne. Mechanizing epistemic logic with Coq. *Annals of Mathematics and Artificial Intelligence*, 48:15–43, 2007.
21. K. Lorenz and P. Lorenzen. *Dialogische Logik*. Darmstadt, 1978.
22. The Coq development team. *The Coq proof assistant reference manual*. LogiCal Project, 2007. Version 8.1.

23. Stephan Merz. Weak alternating automata in Isabelle/HOL. In Mark Aagaard and John Harrison, editors, *TPHOLs*, volume 1869 of *Lecture Notes in Computer Science*, pages 424–441. Springer, 2000.
24. R. Milner. *Communication and concurrency*. Prentice-Hall International Series in Computer Science. Prentice Hall, Inc., 1989.
25. Molière. The middle-class gentleman — wikisource, the free library, 2008. [Online; accessed 19-March-2009].
26. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.
27. Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.
28. B. O'Neill. International escalation and the dollar auction. *J. of Conflict Resolution*, 30(33-50), 1986.
29. M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, Massachusetts, 1994.
30. Martin J. Osborne. *An Introduction to Game Theory*. Oxford, 2004.
31. S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, jun 1992. Springer-Verlag.
32. David Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.
33. R. W. Rosenthal. Games of perfect information, predatory pricing and the chain-store paradox. *Journal of Economy Theory*, 25:92–100, 1981.
34. Stéphane Le Roux. *Abstraction and Formalization in Game Theory*. PhD thesis, École normale supérieure de Lyon (France), January 2008.
35. Martin Shubik. The dollar auction game: A paradox in noncooperative behavior and escalation. *Journal of Conflict Resolution*, 15(1):109–111, 1971.
36. Len Shustek. Interview Donald Knuth: A life's work interrupted. *Communications of the ACM*, 51(8):31–37, August 2008.
37. J. van Benthem. *Logic in Games*. Elsevier, 2006.
38. René Vestergaard. A constructive approach to sequential Nash equilibria. *Inf. Process. Lett.*, 97:46–51, 2006.
39. Rene Vestergaard, Pierre Lescanne, and Hiroakira Ono. Pierre lescanne and hiroakira ono. Research Report JAIST/IS-RR-2006-009, Japan InsT. of Science and Technology, 2006.

A Equalities

Leibniz equality says that $x = y$ if and only if, for every predicate P, $P(x)$ implies $P(y)$. *Extensional equality* says that $f = g$ if and only if, for all x, $f(x) = g(x)$. In general, knowing a (recursive) definition of f and a (recursive) definition of g is not enough to decide whether $f = g$ or $f \neq g$. For instance, no one knows how to prove that the two functions:

$$\begin{aligned} f(1) &= 1 \\ f(2x) &= f(x) \\ f(2x+1) &= f(3x+2). \end{aligned}$$

and

$$g(x) \;=\; 1$$

are equal, despite it is more likely that they are. More generally, there is no algorithm (no rigorous reasoning) which decides whether a given function h is equal to the above function g or not. Thus *extensional equality* is not decidable. Saying that two sequences that have equal elements are equal requires *extensional equality* and it makes sense to reject such an equality when reasoning finitely about infinite objects, like human agents would do.

B Some definition and proposition in Coq vernacular

The notation of functions in the Coq vernacular. In traditional mathematics, the result of applying a function f to the value x is written $f(x)$ and the result of applying f to x and y is written $f(x, y)$, this can be considered as the result of applying f to x then to y and written $f(x)(y)$. In the Coq vernacular, as in type theory, one writes $f\ x$ instead of $f(x)$ and $f\ x\ y$ instead $f(x)(y)$ or instead of $f(x, y)$ and $f\ x\ y\ z$ instead $f(x)(y)(z)$ or $f(x, y, z)$, because this saves parentheses and commas and because functions are everywhere as the core of the formalization. But after all, this is not deep, just a matter of style and Coq accepts syntactic shorthands to avoid these notations when others are desirable.

The definitions The following definition are in the Coq vernacular.

The definition of *History* is

```
CoInductive History: Set :=
— HNil: History
— HCons: Choice → History → History.
```

The definition of *bisimilarity* in the Coqis (Notice the use of the notation $(HCons\ a\ h)$ for $HCons(a)(h)$):

```
CoInductive hBisimilar: History →History → Prop :=
— bisim_HNil: hBisimilar HNil HNil
— bisim_HCons: ∀ (a:Choice)(h h':History),
  hBisimilar h h' → hBisimilar (HCons a h) (HCons a h').
```

The definition of *Game* is:

```
CoInductive Game : Set :=
— gLeaf: Utility_fun → Game
— gNode : Agent → Game → Game → Game.
```

The definition of *Strategy* is:

```
CoInductive Strategy : Set :=
— sLeaf : Utility_fun → Strategy
— sNode : Agent → Choice → Strategy → Strategy → Strategy.
```

The function *s2u* is defined as

```
CoInductive s2u : Strategy → Agent → Utility → Prop :=
— s2uLeaf: ∀ a f, s2u (≪f≫) a (f a)
— s2uLeft: ∀ (a a':Agent) (u:Utility) (sl sr:Strategy),
    s2u sl a u → s2u (≪a',l,sl,sr≫) a u
— s2uRight: ∀ (a a':Agent) (u:Utility) (sl sr:Strategy),
    s2u sr a u → s2u (≪a',r,sl,sr≫) a u.
```

LeadstoLeaf is an **inductive**.

```
Inductive LeadsToLeaf: Strategy → Prop :=
— LtLLeaf: ∀ f, LeadsToLeaf (≪f≫)
— LtLLeft: ∀ (a:Agent)(sl: Strategy) (sr:Strategy),
    LeadsToLeaf sl → LeadsToLeaf (≪a,l,sl,sr≫)
— LtLRight: ∀ (a:Agent)(sl: Strategy) (sr:Strategy),
    LeadsToLeaf sr → LeadsToLeaf (≪a,r,sl,sr≫).
```

Two lemmas about *LeadsToLeaf* and *s2u*:

- $\forall a\ s, LeadsToLeaf\ s \rightarrow \exists u : Utility, s2u\ s\ a\ u.$
- $\forall a\ u\ v\ s, LeadsToLeaf\ s \rightarrow s2u\ s\ a\ u \rightarrow s2u\ s\ a\ v \rightarrow u = v.$

The predicate which we called *"always leads to a leaf"* is formally defined in the vernacular as follows:

```
CoInductive AlwLeadsToLeaf: Strategy → Prop :=
— ALtLeaf : ∀ (f:Utility_fun), AlwLeadsToLeaf (≪f≫)
— ALtL : ∀ (a:Agent)(c:Choice)(sl sr:Strategy),
    LeadsToLeaf (≪a,c,sl,sr≫) → AlwLeadsToLeaf sl →AlwLeadsToLeaf sr →
    AlwLeadsToLeaf (≪a,c,sl,sr≫).
```

Bisimilarity of *strategies* is defined **coinductively**:

```
CoInductive sBisimilar: Strategy → Strategy → Prop :=
— bisim_sLeaf: ∀ f, sBisimilar (≪f≫) (≪f≫)
— bisim_sNode: ∀ a c sl sl' sr sr',
  sBisimilar sl sl' → sBisimilar sr sr' → sBisimilar (≪a,c,sl,sr≫) (≪a, c,sl',sr'≫).
```

SGPE is defined **coinductively** as follows:

```
CoInductive SGPE: Strategy → Prop :=
— SGPE_leaf: ∀ f:Utility_fun, SGPE (≪f≫)
— SGPE_left: ∀ (a:Agent)(u v: Utility) (sl sr: Strategy),
    AlwLeadsToLeaf (≪a,l,sl,sr≫) →
    SGPE sl → SGPE sr →
    s2u sl a u → s2u sr a v → (v ≼u) →
    SGPE (≪a,l,sl,sr≫)
— SGPE_right: ∀ (a:Agent) (u v:Utility) (sl sr: Strategy),
    AlwLeadsToLeaf (≪a,r,sl,sr≫) →
    SGPE sl → SGPE sr →
    s2u sl a u → s2u sr a v → (u ≼v) →
```

SGPE (≪*a*,*r*,*sl*,*sr*≫).

Lemma *AlwLeadsToLeaf_Preservation*: ∀ (*a*:*Agent*)(*s s'*:*Strategy*),
AlwLeadsToLeaf s → *s* $\vdash^{a}\dashv$*s'* → *AlwLeadsToLeaf s'*.

Definition *NashEq* (*s*: *Strategy*): *Prop* :=
∀ *a s' u u'*, *s'*$\vdash^{a}\dashv$*s* →
LeadsToLeaf s' → (*s2u s' a u'*) →
LeadsToLeaf s → (*s2u s a u*) →
(*u'* ⪯*u*).