Extensive Infinite Games and Escalation, an exercise in Agda

Pierre Lescanne

To cite this version:

Pierre Lescanne. Extensive Infinite Games and Escalation, an exercise in Agda. 2020. hal-02874762

HAL Id: hal-02874762
https://hal.archives-ouvertes.fr/hal-02874762
Preprint submitted on 19 Jun 2020

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.
Extensive Infinite Games and Escalation, an exercice in Agda

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

Escalation in games is when agents keep playing forever. Based on formal proofs we claim that if agents assume that resource are infinite, escalation is rational.

Keywords: extensive game, infinite game, sequential game, escalation coinduction, Agda, proof assistant, formal proof.

1 Introduction

Escalation in games is the phenomenon where agents keep playing (or betting if the game consists in bets) forever, leading to their ruin. Since Shubik people claim that such an attitude is not rational. Based on formal proofs we are able to refute such a claim and to say that if agents assume that resource are infinite, escalation is rational. Since our first work which took place before the 2008 financial crisis, evidence show that stating the rationality of escalation makes sense. The only solution for avoiding escalation is then to assume that resource are finite.

In previous works we used an approach based on Coq and coinduction (a dual of induction aimed at reasoning on infinite data structures). Especially in we used dependent types together with coinduction. In this paper, we use coinduction in Agda, because it allows a terse style closed to this of mathematicians. Agda is a formal proof computer environment as well as a dependently typed programming language.

Notice other works using proof assistants for proving properties of agents. For instance, Stéphane Le Roux proved the existence of Nash equilibria using Coq and Isabelle. In a somewhat connected area, Tobias Nipkow proved Arrows theorem in HOL. Agda code of this development are available on GitHub.

2 Games and Strategy Profiles

Since we study game theory, let us first define games. A game is either a leaf or a node. A leaf is a assignment to each agent of a Utility (sometime called a payoff). Note that the type of utility depends on the agent (dependent type). A node contains two entities, put in a record: an agent (the agent who has the trait) and a function next which tells the next positions to be played.

1 https://github.com/PierreLescanne/DependentTypesForExtensiveGames-in-Agda
mutual
  Game = ((a : Agent) → Utility a) ⊎ NodeG

record NodeG : Set where
  coinductive
  field
    ag : Agent
    next : Choice → Game

Notice the key word coinductive which shows that we deal with infinite games. The main concept in game theory is this of strategy profiles. Strategy profiles are like games with at each node a choice, which is the choice of the agent who continues the game. In Agda the sum comes with two functions inj₁ and inj₂. In our case, if \( u \) is a utility assignment of type \( ((a : \text{Agent}) \to \text{Utility} a) \) then inj₁ \( u \) is a Game and \( n \) is a NodeG then inj₂ \( n \) is a Game. Strategy profiles are abbreviated StratProf.

mutual
  StratProf = ((a : Agent) → Utility a) ⊎ NodeS

record NodeS : Set where
  coinductive
  field
    ag : Agent
    next : Choice → StratProf
    ch : Choice

We can define the underlying game of a strategy profile

game : (s : StratProf) → Game
game (inj₁ u) = inj₁ u
game (inj₂ n) = inj₂ (gameN n) where
gameN : NodeS → NodeG
  NodeG.ag (gameN n) = ag n
  NodeG.next (gameN n) c = game (next n c)

The underlying game of a leaf (strategy profile) is the same utility assignment, i.e., a leaf (game). For nodes, games are attributed corecursively. Now let us look at another concept. Given two strategy profiles, one may wonder whether they have the same underlying game. This is given by the binary relation \( \approx^{sg} \).

mutual
data _≈^{sg}_ : StratProf → StratProf → Set where
  \approx^{sg}Leaf : u : ((a : Agent) → Utility a) → inj₁ u \approx^{sg} inj₁ u
  \approx^{sg}Node : n n' : NodeS → n \approx^{sg} n' → inj₂ n \approx^{sg} inj₂ n'

record _≈^{sg}_ (n n' : NodeS) : Set where
  coinductive
  field
    is\(\approx^{sg}\) : ag n \equiv ag n' → ((c : Choice) → next n c \approx^{sg} next n' c)
A leaf has the same game as itself, two nodes have the same game if all their “next” strategy profiles have the same games. Notice that we use the symbol \( \circ \) for concepts associated with \textbf{NodeS}, when the concept without \( \circ \) is associated with \textbf{StratProf}. Given a strategy profile, we may want to compute the utility of an agent. This assumes that the path that follows the choices of the agents leads to a leaf. A strategy profile \( s \) with such a property is said convergent, written \( \downarrow s \). This is defined as follows:

\[
\text{mutual}
\]
\[
data \downarrow : \text{StratProf} \to \text{Set} \quad \text{where}
\]
\[
\downarrow \text{Leaf} : u : (a : \text{Agent}) \to \text{Utility} a \to \downarrow (\text{inj}_1 u)
\]
\[
\downarrow \text{Node} : n : \text{NodeS} \to \circ \downarrow n \to \downarrow (\text{inj}_2 n)
\]

\[
\text{record} \circ \downarrow (n : \text{NodeS}) : \text{Set} \quad \text{where}
\]
\[
\text{inductive}
\]
\[
\text{field}
\]
\[
is\downarrow : \downarrow (\text{next } n (\text{ch } n))
\]

Notice that not all the strategy profile are convergent, for instance the strategy profile \( \text{AcBc} \) of Section 4 is not convergent.

We define the utility assignment \( \hat{\text{u}} \) of a convergent strategy profile. \( \hat{\text{u}} \) takes two parameters: a strategy profile \( s \) and a proof that \( s \) is convergent.

\[
\hat{\text{u}} : (s : \text{StratProf}) \to (\downarrow s) \to (a : \text{Agent}) \to \text{Utility} a
\]
\[
\hat{\text{u}} (\text{inj}_1 u) \downarrow \text{Leaf} = u
\]
\[
\hat{\text{u}} (\text{inj}_2 n) (\downarrow \text{Node } p) = \circ \downarrow n p
\]

\[
\circ \hat{\text{u}} : (n : \text{NodeS}) \to (\circ \downarrow n) \to (a : \text{Agent}) \to \text{Utility} a
\]
\[
\circ \hat{\text{u}} n p = \hat{\text{u}} (\text{next } n (\text{ch } n)) (\text{is} \circ \downarrow p)
\]

Subgame perfect equilibria are very interesting strategy profiles. They are strategy profiles in which the choices of the agents are the best. A leaf is always a subgame perfect equilibrium. A node is a subgame perfect equilibrium if the next strategy profile for the choice of the agent is convergent and is a subgame perfect equilibrium, if for any other node which has the same game and whose next strategy profile is also convergent and is a subgame perfect equilibrium, the utility of the agent of the given node is not less than the utility of the agent of this other node. This is defined formally in \textit{Agda} as follows, where we use \( \rightleftharpoons s \) to tell that \( s \) is a subgame perfect equilibrium.

\[
data \rightleftharpoons : \text{StratProf} \to \text{Set} \quad \text{where}
\]
\[
\rightleftharpoons \text{Leaf} : u : (a : \text{Agent}) \to \text{Utility} a \to \rightleftharpoons \text{inj}_1 u
\]
\[
\rightleftharpoons \text{Node} : n n' : \text{NodeS} \to
\]
\[
\circ \rightleftharpoons n n' \to
\]
\[
\rightleftharpoons (\text{next } n (\text{ch } n)) \to
\]
\[
\rightleftharpoons (\text{next } n' (\text{ch } n')) \to
\]
\[
(\hat{\text{u}} (\text{next } n (\text{ch } n)) p (\text{ag } n)) \rightleftharpoons (\hat{\text{u}} (\text{next } n' (\text{ch } n')) p' (\text{ag } n)) \to
\]
\[
\text{inj}_2 n
\]
3 Escalation

We are now interested in strategy profile leading to escalation.

3.1 Good strategy profile

A first property toward escalation is what we call *goodness*. A strategy profile is *good* if at each node, there is a subgame perfect equilibrium with the same game and the same choice.

\[
\text{mutual} \\
\text{data } \odot_- : (s : \text{StratProf}) \to \text{Set} \text{ where} \\
\odot\text{Node} : n : \text{NodeS} \to \odot n \to \odot (\text{inj}_2 n) \\
\text{record } \odot_- (n : \text{NodeS}) : \text{Set} \text{ where} \\
\text{coinductive} \\
\text{field} \\
\text{iso}_\odot : (n' : \text{NodeS}) \to (\text{inj}_2 n') \to n \simeq sg n' \to \text{ch } n \equiv \text{ch } n' \to \\
\odot (\text{next } n \text{ (ch } n))
\]

In other words, this strategy profile is not itself a subgame perfect equilibrium, in particular, it can be non convergent, but each of its choices is dictated by a subgame perfect equilibrium. Goodness can be considered as *rationality* in the choices of the agents. Reader may notice that goodness is of interest only in infinite games, because in a finite game, there is no difference between a good strategy and a subgame perfect equilibrium.

3.2 Divergent strategy profile

Another property of strategy profiles is *divergence*. In a divergent strategy profile, if one follows the choices of the agents, one never gets to a leaf, but, on the opposite, one runs forever. A divergent strategy profile is written \(\uparrow s\). The formal definition in Agda of divergence looks like this of convergence, but the test for divergence is based on a coinductive record and never hits a leaf, therefore there is no \(\uparrow \text{Leaf}\) case.

\[
\text{mutual} \\
\text{data } \uparrow_- : \text{StratProf} \to \text{Set} \text{ where} \\
\uparrow\text{Node} : n : \text{NodeS} \to \uparrow n \to \uparrow (\text{inj}_2 n) \\
\text{record } \uparrow_- (n : \text{NodeS}) : \text{Set} \text{ where} \\
\text{coinductive} \\
\text{field} \\
\text{iso}_\uparrow : \uparrow (\text{next } n \text{ (ch } n))
\]

An *escalation* is a strategy profile which is both *good* and *divergent*.

4 Strategies with two agents and two choices

To build escalating strategy profiles, we consider the case of two agents Alice and Bob and two choices down and right.
We take the natural numbers $\mathbb{N}$ as utility for both agents and for the $\succ$ relation we take the $\succeq$ relation defined as:

\[
data \succeq : \mathbb{N} \to \mathbb{N} \to \text{Set}
data z \succeq z : \text{zero} \succeq \text{zero}
data s \succeq z : n : \mathbb{N} \to \text{suc n} \succeq \text{zero}
data s \succeq s : n \text{ m} : \mathbb{N} \to \text{n} \succeq \text{m} \to \text{suc n} \succeq \text{suc m}
\]

A utility assignment is for instance this which assigns 1 to Alice and 0 to Bob:

\[
uA1B0 : \text{AliceBob} \to \mathbb{N}
uA1B0 \text{ Alice} = 1
uA1B0 \text{ Bob} = 0
\]

from which we can build a leaf strategy profile:

\[
\text{A1B0} : \text{StratProf}
\text{A1B0} = \text{inj}_1 uA1B0
\]

which is convergent.

\[
\downarrow A1B0 : \downarrow A1B0
\downarrow A1B0 = \downarrow \text{Leaf}
\]

From the utility assignment which assigns 0 to Alice and 1 to Bob on can build the convergent strategy profile A0B1.

Moreover, we build an infinite strategy AcBs, in which Alice continues always and Bob stops always:

\footnote{We could have taken a utility with only two values, but we feel that the reader is more acquainted with natural numbers for utilities.}

\footnote{In this case, the type of utility does not depend on the agent.}
We notices that by mutual co-recursion, $AcBs$ is defined together with an infinite strategy profile $BsAc$ which starts with a node of which Bob is the agent. Those strategies are like infinite combs.

With down one reaches always a leaf and with right one goes always to a new strategy profile, which is a node. There is a variant of the node $\circ AcBs$, in which the first choice of Alice is down instead of right.

We prove that $\circ AcBs$ and $Var\circ AcBs$ have the same game. Likewise we prove that $AcBs$ is convergent i.e., $\downarrow AcBs$. Those two facts are key steps in the proof that $AcBs$ is subgame perfect equilibrium i.e., that $\equiv AcBs$. 

mutual

$AcBs : StratProf$

$AcBs = \text{inj}_2 \circ AcBs$

$\circ AcBs : \text{NodeS}$

$ag \circ AcBs = Alice$

$ch \circ AcBs = right$

$next \circ AcBs down = AOB1$

$next \circ AcBs right = BsAc$

$BsAc : StratProf$

$BsAc = \text{inj}_2 \circ BsAc$

$\circ BsAc : \text{NodeS}$

$ag \circ BsAc = Bob$

$ch \circ BsAc = down$

$next \circ BsAc down = A1B0$

$next \circ BsAc right = AcBs$

We prove that $\circ AcBs$ and $Var\circ AcBs$ have the same game. Likewise we prove that $AcBs$ is convergent i.e., $\downarrow AcBs$. Those two facts are key steps in the proof that $AcBs$ is subgame perfect equilibrium i.e., that $\equiv AcBs$. 

We notices that by mutual co-recursion, $AcBs$ is defined together with an infinite strategy profile $BsAc$ which starts with a node of which Bob is the agent. Those strategies are like infinite combs.
On the same paradigm we built a strategy profile \( AsBc \) in which \( A \) stops and \( B \) continues and which is proved to be convergent and to be a subgame perfect equilibrium. We also build a strategy profile in which \( A \) and \( B \) both continue. 

\[
\text{mutual} \quad AcBc : \text{StratProf} \\
AcBc = \text{inj}_2 \circ AcBc
\]

\[
\circ AcBc : \text{NodeS} \\
ag \circ AcBc = Alice \\
ch \circ AcBc = right \\
next \circ AcBc \downarrow = A0B1 \\
next \circ AcBc \rightarrow = BcAc
\]

\[
BcAc : \text{StratProf} \\
BcAc = \text{inj}_2 \circ BcAc
\]

\[
\circ BcAc : \text{NodeS} \\
ag \circ BcAc = Bob \\
ch \circ BcAc = right \\
next \circ BcAc \downarrow = A1B0 \\
next \circ BcAc \rightarrow = AcBc
\]

\( AcBs, AcBc \) and \( AsBc \) have the same game. Unlike \( AcBs \) and \( AsBc \), the strategy profile \( AcBc \) is divergent, i.e., \( \uparrow AcBc \). Moreover \( AcBc \) is good which means \( \circ AcBc \).

5 Conclusion

Since \( AcBc \) is good and divergent, \( AcBc \) is an escalation. Hence we proved formally the claim of the introduction, namely if agents assume that resource are infinite, escalation is rational.

In the current implementation, the type of choices is the same for all the agents. However, one may imagine that this type may depend on the agents. Making the type of choices depending on the agents is object of the current investigation.

References