JAIST Repository
https://dspace.jaist.ac.jp/

Title	The inductive and modal proof theory of Aumann's theorem on rationality
Author(s)	Vestergaard, Rene; Lescanne, Pierre; Ono, Hiroakira
Citation	Research report (School of Information Science, Japan Advanced Institute of Science and Technology), IS-RR-2006-009: 1-17
Issue Date	2006-07-07
Type	Technical Report
Text version	publisher
URL	http://hdl.handle.net/10119/8411
Rights	
Description	リサーチレポート（北陸先端科学技術大学院大学情報科学研究科）

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

The Inductive and Modal Proof Theory of Aumann's Theorem on Rationality

René Vestergaard
JAIST, Japan

Pierre Lescanne
ENS, Lyon, France

Hiroakira Ono
JAIST, Japan

IS-RR-2006-009

July 7, 2006

Abstract

Aumann's Theorem on Rationality is a well-know but also contentious result in economics. It pertains to sequential game theory and says that "common knowledge of rationality leads to backwards-induction equilibria".[1] We present a formalist analysis of the result in a meta-theory with primitive support for proof and definition by induction. The analysis shows in part that validity of the result can be reduced to a so-called modal axiom T. Complementing the particular axiom T of Aumann's set-up, we propose an alternative axiom that results in "decidable (local) rationality leads to backwards-induction equilibria". Aumann's result follows from ours but not vice versa and the two axioms T appear to be independent. Our development has been verified in full detail and fully transparently in the Coq proof assistant.

The first part of the article is written as a brief overview of the theory behind formal, inductive, and proof-theoretic reasoning in a mechanised proof assistant, such as Coq. It is our contention that such tools go well with sequential game theory and we advocate more wide-spread usage.

[1] The result is part of Aumann's research program on the benefits of sustained involvement in conflict resolution for which he was co-awarded the 2005 Nobel Memorial prize in economics.

1 Introduction

Aumann's Theorem on Rationality is stated in [2] and is further discussed, e.g., in [9, 18, 21]. It says that if it is "established wisdom" among the players in a sequential game that they behave *rationally* then the considered *strategy profile* is a *backwards-induction Nash equilibrium* [15, 16, 19, 20]. The existing presentations are by pen-and-paper and are model-theoretic in nature; they differ in how they express "established wisdom": as the common knowledge modality [2] or a refinement there-of [18]. The result is not uniformly considered to hold [9, 21].

1.1 Our Contribution

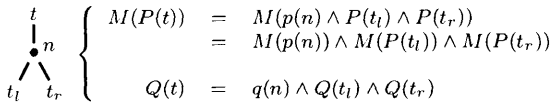
We undertake a formal(ist) development and analysis of Aumann's result. To rule out the possibility of omissions and to guarantee transparency, everything has been verified in the Coq proof assistant [5]. Coq is an LCF-style interactive theorem prover, implying that proofs are first-class objects that have been constructed by and can be used for computational reasoning over inductive structures. Our approach is proof-theoretic, rather than model-theoretic, and allows us to analyse aspects of the result that do not appear to have been addressed so far. This concerns specific parts of the axiomatic requirements on "wisdom" and rationality and we are ultimately lead to consider the question of decidability of the decisions that the players are supposed to be rational in making according to Aumann's set-up.

The novel aspect of our formal development that contributes most to the already-established understanding of the problem is our use of inductive definitions and reasoning. While the common-knowledge modality and its refinements are defined as least fixed-points and, thus, are inductively structured, we take advantage of a more basic notion of inductive structure that typically does not exist model-theoretically, namely of the games themselves [23]. Using induction, the proof of our main result is a mere couple of handfuls of basic and formal steps.[2]

[2]To be exact, the proof is 11 possibly-parallel steps or 14 atomic steps, see Figure 8.

1.2 This Article

Our approach is based on the idea of treating the intuitive form of sequential games, i.e., trees, as formal objects in their own right. Abstractly speaking, Aumann's theorem says that it is the case for all trees that if one particular (composed) property, $M(P(t))$, holds of a tree, t, then so does another one, $Q(t)$. Moreover, P and Q are defined as the conjunctions of properties $p(n)$ and $q(n)$ for each node, n, and M distributes over conjunction.



We first observe that proving that $M(P(t))$ implies $Q(t)$ can proceed by induction over the tree structure of t: in a proof by induction, we have that $M(P(t_l))$ implies $Q(t_l)$ and $M(P(t_r))$ implies $Q(t_r)$ by induction hypotheses, irrespective of the nature of M, P, and Q, and the real proof burden is therefore to show that $M(p(n))$ implies $q(n)$.[3] Proof by induction works particularly well when the involved properties are *compositional*, as above.

In Section 2, we review the use of type theory to do inductive proofs. In Section 4, we specify all the notions that are required for Aumann's result and briefly review the relevant game theory. In Section 5, we address the original form of Aumann's theorem from an inductive perspective, which sets the stage for Section 7, where we present two different proofs of our modified form of the result. In Section 8, we prove the original Aumann's Theorem on Rationality.

2 Formalism

Our formalisation takes place in type theory and we now review the basic principles. The main product of this section is a gradually built-up provability relation that essentially suffices for our formalisation. The section is neither a tutorial nor a foundational contribution of this paper. For that, refer, e.g., to [5].

[3]We will show in Section 5 that trying to prove that $M(p(n) \wedge P(t_l) \wedge P(t_r))$ implies $q(n)$ in this particular case boils down to proving that $M(p(n))$ implies $q(n)$.

2.1 The Formal Language

Underlying our formalism is a higher-order language that comprises modal predicates with function and relation symbols. Modalities are not usually primitive to type theory but they can be defined on top basically in the same manner that we define them, as we shall see. We treat modalities primitively to be faithful to existing presentations of Aumann's result.

Definition 1 (Symbols and Modalities)

- $\mathcal{V}$, *ranged over by* x, *are* variable names.
- $\mathcal{F}$, *ranged over by* f, *are* function symbols.
- $\mathcal{R}$, *ranged over by* r, *are* relation symbols.
- $\mathcal{M}$, *ranged over by* m, *are* modalities.

All sets are disjoint; $\mathcal{V} \cup \mathcal{F}$ *is ranged over by* n.

Function and relation symbols are non-logical and will be user-definable according to a general inductive scheme that is guaranteed to preserve consistency, see Sections 2.5 and 2.8. Modalities affect the logical meaning of formulas and will be defined on a case-by-case basis, see Section 2.4. Terms, to be defined next, serve the dual role of representing predicates and denoting values. In fact, a predicate is a term that denotes a propositional value, as we shall see.

Definition 2 (Terms)

$$\mathrm{T} \quad ::= \quad \mathcal{V} \mid \mathcal{F} \mid \mathrm{T}\,\mathrm{T}$$

Let t *range over* T.

Definition 3 (Formulas)

$$\begin{array}{lll} \mathrm{P} & ::= & \mathrm{P} \wedge \mathrm{P} \mid \mathrm{P} \vee \mathrm{P} \mid \mathrm{P} \supset \mathrm{P} \mid \mathtt{F} \\ & & \mid \mathrm{T} \mid \mathcal{R}(\mathrm{T},\mathrm{T}) \\ & & \mid \mathcal{M}(\mathrm{P}) \end{array}$$

Let p *range over* P; *write* $\neg p$ *for* $p \supset \mathtt{F}$ *and* T *for* $\neg\mathtt{F}$.

Examples of formulas are $\mathrm{T} \supset \mathtt{F}$, $x \vee \neg x$, with $x \in \mathcal{V}$, and using function and relation symbols that we define later, *Succ t* and *LessEq*(*Zero*, *Succ t*). Needless to say, and as implied, not all formulas are provable.

2.2 Well-Sortedness

Before addressing provability, we make terms and formulas well-sorted. We do it relative to a set of user-defined base sorts and the constant EPROP; EPROP is the sort of propositional values, i.e., the subset of our formulas that are candidates for being proved.

Definition 4 (Sorts) *Let* U *be a set of* user-defined base sorts.

$$\mathrm{S} \quad ::= \quad U \mid \textsc{eProp} \mid \mathrm{S} \to \mathrm{S}$$

Let s *range over* S.

Definition 5 (Well-Sortedness) *Let* Δ *range over pairs* $\langle \Delta_t, \Delta_f \rangle$, *where* Δ_t *ranges over finite functions from* $\mathcal{V} \cup \mathcal{F}$ *to* S *and* Δ_f *ranges over finite functions from* $\mathcal{R}$ *to* $\mathrm{S} \times \mathrm{S}$;[4] *let* $\oplus$ *range over* $\{\wedge, \vee, \supset\}$. *Define* $\rhd_t$, well-sortedness for terms, *and* $\rhd_f$, well-sortedness for formulas, *as follows.*

$$\frac{\Delta_t \rhd_t t_f : s_a \to s_r \quad \Delta_t \rhd_t t_a : s_a}{\Delta_t \rhd_t t_f\ t_a : s_r} \qquad \frac{}{\Delta_t \rhd_t n : s} \ \text{if } \Delta_t(n) = s$$

$$\frac{\Delta \rhd_f p}{\Delta \rhd_f m(p)} \qquad \frac{\Delta \rhd_t t_1 : s_1 \quad \Delta \rhd_t t_2 : s_2}{\Delta \rhd_f r(t_1, t_2)} \ \text{if } \Delta_f(r) = \langle s_1, s_2 \rangle$$

$$\frac{\Delta \rhd_f p_1 \quad \Delta \rhd_f p_2}{\Delta \rhd_f p_1 \oplus p_2} \qquad \frac{}{\Delta \rhd_f \mathtt{F}} \qquad \frac{\Delta \rhd_t t : \textsc{eProp}}{\Delta \rhd_f t}$$

The sorting contexts, Δ, list arities and sorts of the various symbols and variables we are allowed to use. The rules above say, for example, that F is a well-sorted formula in any sorting context: $\Delta \rhd_f \mathtt{F}$. The lower-right rule says that a term that is EPROP-sorted may be used as a formula, i.e., it is a predicate.

2.3 Propositional Provability

We now begin to incrementally define our provability relation. The first component is propositional logic.

Definition 6 *Let* Γ *range over lists of formulas,* P, *and let* Δ *be as defined in Definition 5. The propositional part of our provability relation,* $\vdash^{\Delta}$, *is given in Figure 1. We write* ε *for empty* Γ *and we say that* p *is a* $\vdash^{\Delta}$*-theorem if* $\varepsilon \vdash^{\Delta} p$.

[4] Formally, Δ should be set up as a dependently typed list.

$$\frac{}{\Gamma_1, p, \Gamma_2 \vdash^{\Delta} p}\,(Assm) \qquad \text{if } \Delta \rhd_f p$$

$$\frac{\Gamma \vdash^{\Delta} p_l \quad \Gamma \vdash^{\Delta} p_r}{\Gamma \vdash^{\Delta} p_l \wedge p_r}\,(I_\wedge)$$

$$\frac{\Gamma \vdash^{\Delta} p_l \wedge p_r}{\Gamma \vdash^{\Delta} p_l}\,(E^l_\wedge) \qquad \frac{\Gamma \vdash^{\Delta} p_l \wedge p_r}{\Gamma \vdash^{\Delta} p_r}\,(E^r_\wedge)$$

$$\frac{\Gamma \vdash^{\Delta} p_l \vee p_r \quad \Gamma, p_l \vdash^{\Delta} p \quad \Gamma, p_r \vdash^{\Delta} p}{\Gamma \vdash^{\Delta} p}\,(E_\vee)$$

$$\frac{\Gamma \vdash^{\Delta} p_l}{\Gamma \vdash^{\Delta} p_l \vee p_r}\,(I^l_\vee) \qquad \frac{\Gamma \vdash^{\Delta} p_r}{\Gamma \vdash^{\Delta} p_l \vee p_r}\,(I^r_\vee)$$

$$\frac{\Gamma, p_a \vdash^{\Delta} p_b}{\Gamma \vdash^{\Delta} p_a \supset p_b}\,(I_\supset) \qquad \text{if } \Delta \rhd_f p_a$$

$$\frac{\Gamma \vdash^{\Delta} p_a \supset p_b \quad \Gamma \vdash^{\Delta} p_a}{\Gamma \vdash^{\Delta} p_b}\,(E_\supset) \qquad \frac{\Gamma \vdash^{\Delta} \mathtt{F}}{\Gamma \vdash^{\Delta} p}\,(E_{\mathtt{F}})$$

Fig. 1. Propositional provability

Rule $(I_\wedge)$, for example, says that we can prove a conjunction in some contexts, Γ, Δ, by proving each conjunct in the same contexts. Rule $(I_\supset)$ says that we can prove an implication by proving the conclusion and then discharging the assumption. The rules for the connectives come in introduction, elimination pairs, except for $\mathtt{F}$, which we cannot introduce, and they are entirely standard (but see Section 6). We note that the side-conditions in the figure serve to guarantee that the considered formulas are built exclusively from EPROP-sorted entities. An example of a propositional proof is as follows.

Proposition 7 $\varepsilon \vdash^{\Delta} \mathtt{T}$, *for any* Δ.

Proof

$$\frac{\dfrac{}{\mathtt{F} \vdash^{\Delta} \mathtt{F}}\,(Assm)}{\varepsilon \vdash^{\Delta} \mathtt{F} \supset \mathtt{F}}\,(I_\supset)$$

□

$$\frac{}{\Gamma \vdash^{\Delta} K_a(p) \supset p}\,(T_{K_a})$$

$$\frac{\varepsilon \vdash^{\perp} p}{\varepsilon \vdash^{\perp} K_a(p)}\,(Gen_{K_a})$$

$$\frac{}{\Gamma \vdash^{\Delta} (K_a(p_1 \supset p_2)) \supset K_a(p_1) \supset K_a(p_2)}\,(K_{K_a})$$

Fig. 2. The knowledge modality, $K_a(-)$, for agent a

$$\frac{}{\Gamma \vdash^{\Delta} C_G(p) \supset p \wedge E_G(C_G(p))}\,(Fix)$$

$$\frac{\varepsilon \vdash^{\perp} p_0 \supset p \wedge E_G(p_0)}{\varepsilon \vdash^{\perp} p_0 \supset C_G(p)}\,(Least)$$

Fig. 3. The common-knowledge modality, $C_G(_)$

2.4 Epistemic Provability

Aumann's result involves statements about the knowledge of the considered group of agents. Formally, this is done with so-called epistemic logic [10]. The basis of epistemic logic is the modality K_a that is intended to capture that "agent a knows that", while two derived modalities address the situations of several agents sharing knowledge and of shared knowledge of shared knowledge *ad infinitum*.

Definition 6 (cont'd) *Let* $\perp$ *be the no-where defined* Δ *and let* G *be some group (read: finite set) of* agents. *The* knowledge modality, $K_a(-)$, *for agent* a *is defined in Figure 2 and the* common-knowledge modality, $C_G(_)$, *is defined in Figure 3, with the* shared-knowledge modality, $E_G(-)$, *defined thus:*

$$E_G(p) \quad \triangleq \quad \bigwedge_{a \in G} K_a(p)$$

Rule (Gen_{K_a}) is called *knowledge generalisation* and it coincides with the usual modal *necessitation* rule. The rule implies that agents are able to pick up any bit of knowledge as long as it is provable. The modality $C_G(p)$ says that p, $E_G(p)$, $E_G(E_G(p))$, and

more generally $E_G^n(p)$ hold for any $n \in \mathbb{N}$, where E_G^n is n-iterated E_G. In other words, we can and do define $C_G(p)$ as the least fixed point of the following equation [14].

$$x \quad \Leftrightarrow \quad p \wedge E_G(x).$$

The equation should be read to say that, e.g., $C_G(p)$ holds if and only if p holds and $C_G(p)$ is shared knowledge. A solution to an equation of the form $x = F(x)$ is called a *fixed point* of F. More precisely, the rules in Figure 3 ultimately stipulate that $C_G(p)$ is the *least* fixed point of the function $x \mapsto p \wedge E_G(x)$, where equality is logical equivalence, $\Leftrightarrow$, and "least" is taken w.r.t. the order induced by implication: a proposition p_1 is *less than* a proposition p_2 if $p_2 \supset p_1$.

With this, we note that C_G satisfies the same properties as K_a; the proofs are verified elsewhere [14].

Lemma 8

$$\frac{}{\Gamma \vdash^{\Delta} C_G(p) \supset p}\,(T_C) \qquad \frac{\varepsilon \vdash^{\perp} p}{\varepsilon \vdash^{\perp} C_G(p)}\,(Gen_C)$$

$$\frac{}{\Gamma \vdash^{\Delta} (C_G(p_1 \supset p_2)) \supset C_G(p_1) \supset C_G(p_2)}\,(K_C)$$

2.5 Inductively Defined Sorts

The sorts and symbols that index our formalism are used to define objects-of-interest inductively [1, 22]. An inductive definition amounts to a least fixed-point construction, which implies that all defined objects are well-founded. For example, the natural numbers can be inductively defined as either zero or the successor of another natural number.

$$\text{Nat} ::= \text{Zero} \mid \text{Succ(Nat)}$$

The definition introduces the new sort Nat as well as the (nullary) function symbol Zero of sort Nat and the (unary) function symbol Succ of sort Nat $\rightarrow$ Nat. The type-theoretic way of guaranteing well-definedness is to require that all recursive occurrences of the defined sort are non-negative in the types of the listed constructors, i.e., of the introduced function symbols [1]. Informally speaking, this means

$$\frac{\Gamma \vdash^{\Delta} \mathrm{P}\ \mathit{Zero} \qquad \Gamma \vdash^{\Delta, x:\mathit{Nat}} (\mathrm{P}\ x) \supset \mathrm{P}\ (\mathit{Succ}\ x)}{\Gamma \vdash^{\Delta} \mathrm{P}\ n}\,(\mathit{sInd}^{\mathrm{P}}_{\mathit{Nat}})$$

if $\Delta_t(\mathit{Zero}) = \mathit{Nat}, \Delta_t(n) = \mathit{Nat}, \Delta_t(\mathit{Succ}) = \mathit{Nat} \rightarrow \mathit{Nat}$

Fig. 4. Structural induction over *Nat* for P

that no constructor may take an argument that, in the simplest case, takes the constructed domain as an argument: $(\mathit{Nat} \rightarrow \mathit{Nat}) \rightarrow \mathit{Nat}$ is forbidden, for example, because of the left-most *Nat*.

By construction, an algebraic structure defined in the style of *Nat* comes equipped with a (sound) proof principle called *structural induction*, which, in the case of *Nat*, is weak number induction, see Figure 4.

Definition 6 (cont'd) *For Inductively defined sorts, include* structural induction *rules, see Figure 4 in the case of Nat, in the provability relation.*

We note that $\Delta, x : \mathit{Nat}$ means that Δ_t is extended with x of sort *Nat*; Γ, Δ are assumed not to reference x. In general, Δ is similarly extended with (fresh) variables for the local parts of all cases. We also note that it is implicit in Figure 4 that $\Delta_t \rhd_t \mathrm{P} : \mathit{Nat} \rightarrow \textsc{eProp}$ and that structural induction should be thought of as concluding $\forall n : \mathit{Nat}.\, \mathrm{P}\ n$, although we have suppressed universal quantification in the presentation. Structural induction says that a predicate, P, will hold for all elements in an inductively defined set if all the considered constructors preserve the property. Informally, structural induction is sound because, e.g., all *Nat*s i) are finite in size and ii) have an outermost constructor, implying that any *Nat* is covered by the premises in an effective manner.

Another example of an inductive definition is *binary trees*.

$$\text{bTree} ::= \text{nil} \mid \text{bTree} \bullet \text{bTree}$$

The informal version of structural bTree-induction reads as follows.

$$\frac{\Gamma \vdash^{\Delta} p_0}{\Gamma \vdash^{\Delta} p}\,(comp) \qquad \text{if } \downarrow p = p_0 \text{ and } \Delta \rhd_f p$$

Fig. 5. Computational reasoning

$$\frac{P(\mathrm{nil}) \qquad P(bT_1) \wedge \mathrm{P}(bT_2) \supset \mathrm{P}(bT_1 \bullet bT_2)}{\forall bT \in \mathrm{bTree}\,.\,\mathrm{P}(bT)}$$

2.6 Structural-Recursive Functions

One way to define a predicate is as a function into EPROP, as we saw above. If such a function were to be undefined for any (sort-correct) argument, our theory could be subject to an inconsistency. The reason is that propositions and types, and provability and type inhabitation coincide [11]. Fortunately, a close cousin of structural induction amounts to a schema by which we can define total, computable functions and we accept as well-defined any function symbol that has been defined by case-splitting on an inductive structure provided all recursive calls are made with a case-given sub-structure of the principal argument. This is called *structural recursion.* In order to decide whether a natural is even, we can therefore define the following function by cases, using a recursive call on n in the Succ-case.

$$\begin{aligned} \mathrm{IsEven(Zero)} &= \mathtt{T} \\ \mathrm{IsEven(Succ}(n)) &= \neg\mathrm{IsEven}(n) \end{aligned}$$

Informally, IsEven, i) is *functional*, i.e., is a relation that is not one-to-many, because the case-splitting is non-overlapping, ii) is *computable* because all recursive calls are made on a sub-structure of a well-founded element (in Nat), and iii) is total (on Nat) because the case-splitting also is exhaustive. If we attempt to prove (IsEven Zero) we would like to succeed because it computes to the provable T. To accomplish this formally, we add the following conditional rule to our unfolding type theory.

Definition 6 (cont'd) *Let the* computational reasoning *rule be as given in Figure 5, with* $\downarrow$ *doing "β-normalisation" (i.e., exhaustive definition unfolding)*

$$\frac{}{\Gamma \vdash^{\Delta} \mathrm{LessEq}(t,t)}\,(rInd_{\mathrm{LessEq}^1}) \qquad \text{if } \Delta_t \rhd_t t : \mathrm{Nat}$$

$$\frac{\Gamma \vdash^{\Delta} \mathrm{LessEq}(t_1,t_2)}{\Gamma \vdash^{\Delta} \mathrm{LessEq}(t_1,\mathrm{Succ}(t_2))}\,(rInd_{\mathrm{LessEq}^2})$$

$$\text{if } \Delta_t(\mathrm{LessEq}) = \langle \mathrm{Nat}, \mathrm{Nat} \rangle$$

Fig. 6. Ad hoc rule induction for LessEq

of structural-recursive functions relative to any outermost constructors, e.g., Zero, *in their arguments.*

Note that p in Figure 5 can be a t, by definition, i.e., it can be a predicate. Note also that the $\downarrow p = p_0$ side-condition is decidable because it involves structural recursion, only. I.e., it is decidable whether occurrences of the (*comp*)-rule are correct.[5] For suitable Δ, we thus have the desired result because, in fact, $\downarrow$ (IsEven(Zero)) = T.

$$\cfrac{\cfrac{\cfrac{}{\mathtt{F} \vdash^{\Delta} \mathtt{F}}\,(Assm)}{\varepsilon \vdash^{\Delta} \mathtt{T}}\,(I_{\supset})}{\varepsilon \vdash^{\Delta} \mathrm{IsEven(Zero)}}\,(comp)$$

2.7 Relations

An alternative and more general way of defining predicates is to use an ad hoc inductive form. In this presentation, we reserve the form for (binary) relations.

$$\frac{}{\mathrm{LessEq}(t,t)} \qquad \frac{\mathrm{LessEq}(t_1,t_2)}{\mathrm{LessEq}(t_1,\mathrm{Succ}(t_2))}$$

Because we only consider relations, which always and implicitly are of EPROP sort when supplied with well-sorted arguments, the definitions translate directly into new proof rules of the formalism.

[5] The difference between a side-condition and a premise of a rule is that an occurrence of the latter is equipped with a proof, namely the tree that ends there, while side-conditions belong at the meta-level.

Definition 6 (cont'd) *For relation symbols, add the defining rules as proof rules, see Figure 6, in the case of* LessEq.

Proving that, e.g., LessEq holds using rules such as those in Figure 6 is called *rule induction*. The free form of the involved rules does not prescribe well-foundedness in general, which is why we treat them "internally" in the proof system, rather than "externally" in the term language the way we did for structural-recursively defined predicates. Having the rules "internally" guarantees that any use of them will be in a well-founded proof tree.

2.8 Meta-Theory

Although we will not attempt to prove consistency of the outlined type theory, we will now discuss the pertinent issues. For the interested reader, we note that our formalism has been constructed to be a subset, e.g., of the calculus of inductive constructions [4, 17].

Theorem 9 $\varepsilon \not\vdash^{\Delta} \mathtt{F}$, *for any* reasonable[6] Δ.

Informally, we have consistency because the propositional part, see Figure 1, is consistent in its own right; the modal part, see Figures 2 and 3, conservatively extends the propositional part; structural induction, see Figure 4, is sound by construction as discussed; computational reasoning, see Figure 5, amounts to complicated ways of expressing already-provable properties; and rule induction, see Figure 6, is a conservative extension to the theory, because it involves novel symbols, like in the case of modalities.

Abstractly speaking, formal consistency arguments for extensible type theories typically proceed by the formulation of a more powerful type theory that allows (in a non-extended way) for the definition of functions that sends the various types of definitions we have discussed to their associated proof principles. Consistency of the extensible framework therefore follows from the fixed one.

[6] Reasonable, e.g., means obeying the constraints of dependent typing [4, 17].

3 Reasoning with Coq

The previous section is a generic description of the underlying theory of a number of formal proof assistants (that, moreover, is implementable in many more). As it happens, the present development has been undertaken in Coq [5] and the vernacular we use reflects this. However, the use of Coq is not an essential requirement. Still, and as we show in this section, Coq can directly accommodate Section 2. First, we indicate that we use Coq's *Set* for our U.

Definition $U := Set.$

Secondly, we can inductively define the new sort Nat with Coq notation that makes the sorts clear.

Inductive $Nat : U :=$
— $Zero : Nat$
— $Succ : Nat \rightarrow Nat.$

As mentioned earlier, this inserts *Nat* into U and makes *Zero* and *Succ* available as function symbols in Δ (with the indicated sorts). Behind the scenes, it also makes available structural induction and recursion for *Nat*. In the (Coq-)definition below, the keyword Fixpoint indicates that we are trying to define yet another function symbol, *IsEven*, that will take argument $n : Nat$. In order to establish that *IsEven* is well-defined, we indicate that we justify the definition by structural recursion on n: $\{struct\ n\}$. This implies that we are allowed to call *IsEven* on *n0* in the *Succ*-case of the case-splitting on n.

Proposition 10 *IsEven, defined below, is a total, computable function on Nat.*

Fixpoint IsEven (n : Nat) {struct n} : Prop :=
match n with
— *Zero* ⇒ *True*
— *Succ n0* ⇒ ¬ *(IsEven n0)*
end.

Following this brief introduction to Coq syntax, we note that we leave a number of the things discussed in Section 2 to Coq, in particular universal quantification, and management of the contexts, Γ, Δ and their use in well-sorting. We also let Coq create structural and rule induction principles for us, as well as the

necessary arguments for enabling definition by structural recursion (through Fixpoint). Finally, we rely on Coq for terms and computational reasoning. For the rest, and for transparency, we use Coq's object level, as we shall see next.

4 Formalisation

We will now formally define all the concepts that are relevant to the statement of Aumann's theorem, beginning with the modal and propositional part of the language of formulas that Aumann's theorem is expressed in, see Definition 3. The game theory part follows [23]. The language of formulas is indexed by some set of agents, G (with equality), for the epistemic modality K; we implicitly assume that the epistemic modality C is with respect to all of G.

Coq-Formalism 11 (Formulas)
Variable $G : U$.
Variable $agentEq : G \to G \to bool$.
Axiom $agentEqual : \forall\ a,\ (agentEq\ a\ a) = true$.

Inductive $eProp : U :=$
— $eTrue : eProp$
— $Neg : eProp \to eProp$
— $Imp : eProp \to eProp \to eProp$
— $And : eProp \to eProp \to eProp$
— $Or : eProp \to eProp \to eProp$
— $K : G \to eProp \to eProp$
— $C : eProp \to eProp$.

For convenience, we shall use short-hand notation for a particular combination of logical connectives, as well as allowing ourselves to use standard-looking in-fix forms of some of the connectives.

Coq-Formalism 12 (Notation)
Definition $nKn\ (a{:}G)\ (P{:}eProp) := Neg\ (K\ a\ (Neg\ P))$.

Infix "$\Longrightarrow$" $:= Imp$ *(right associativity, at level 85).*
Infix "&&" $:= And$ *(left associativity, at level 50).*
Infix "vv" $:= Or$ *(left associativity, at level 50).*

Instructions like (*right associativity, at level* 85) are on-the-fly defined parsing-directives that disambiguate, e.g., $A \Longrightarrow B \Longrightarrow C$ to mean $A \Longrightarrow (B \Longrightarrow C)$; the number indicates the priority for the application of such rules. Suffice it to say that the parsing-directives above facilitate the standard short-hands.

4.1 Basics of Game Theory

Our interest in games is how they are being played and how they end with players either winning, losing, or making even, relative to some notion of payoff. We first formalise end-situations by introducing the following primitive sort, function symbol (for less-than-or-equal-to on the payoff values), and short-hand name for a composed sort.

Coq-Formalism 13 (Basics)
Variable $payoff : U$.
Variable $eLeq : payoff \to payoff \to eProp$.
Definition $payoffs := G \to payoff$.

For the playing part, we note that Aumann's theorem pertains to sequential games in which players choose between their available options in some play-specific order. For convenience and readability, we shall restrict attention to cases where an agent always has exactly two options.

Coq-Formalism 14
Inductive $choice : U :=$
— $lchoice$
— $rchoice$.

This is not a limitation because Aumann's theorem is equivalent whether stated for the binary case or the case where an agent can have one or more options available to him. It is worth noting that while *choice* is defined Inductively, it is inductive in the trivial sense of being defined point-wise: *choice* is the two-point set consisting of constants (read: nullary function symbols) *lchoice* and *rchoice*.

4.2 Strategies, Inductively

To state and prove Aumann's theorem, we need only consider strategies, i.e., games in which each player has decided (up-front, if you wish) how to choose whenever it is his turn. Sequential games (in extensive form, as considered by Aumann et al) are trees

where the internal nodes formalise the options available to the player-at-turn at that particular juncture and where reaching a leaf marks the end of a play of the game. In other words, in order to formalise (binary) strategies, we simply re-use the definition of (binary) trees given in Section 2.5 and, in addition, annotate leafs with payoff functions and internal nodes with the relevant agent owner and his strategy-determining choice.

Coq-Formalism 15 (Binary Strategies)

```
Inductive strategy : U :=
— sLeaf : payoffs → strategy
— sNode : G
          → choice
          → strategy → strategy
          → strategy.
```

The structural induction principle for *strategy* essentially coincides with bTree's because the extra annotations do not affect the structure of the defined objects.

$$\frac{\mathrm{P}(sLeaf\ po) \qquad \mathrm{P}(s_1) \wedge \mathrm{P}(s_2) \supset \mathrm{P}(sNode\ a\ c\ s_1\ s_2)}{\forall s \in strategy\,.\,\mathrm{P}(s)}$$

Structural *strategy*-induction is used in and, in fact, carries most of our proofs.

4.3 Payoffs, Recursively

The *payoffs* induced by the indicated choices in a *strategy* can be computed by structurally-recursively calling a function on the chosen sub-strategy in internal nodes and, ultimately, returning the encountered payoff function.

Coq-Formalism 16 (Induced Pay-offs)

```
Fixpoint stratPO (s:strategy) {struct s} : payoffs :=
match s with
— (sLeaf po) ⇒ po
— (sNode a c sl sr)
    ⇒ match c with
        — lchoice ⇒ (stratPO sl)
        — rchoice ⇒ (stratPO sr)
        end
end.
```

Proposition 17 *stratPO is a total, computable function.*

Proof By construction. □

4.4 Equilibrium Predicate

A Nash equilibrium is a strategy in which no agent can change one or more of his choices to generate a better overall result for himself, in the sense of *stratPO*. Aumann's theorem predicts that common knowledge of rational decision-making results in a particular kind of Nash equilibrium [15, 16], called backwards induction (in stand-alone form, due to [19, 20]). Backwards induction is characterised by locally-enforced optimality of decisions, i.e., we can define a predicate for backwards induction by structural recursion.

Coq-Formalism 18

```
Fixpoint eBI (s:strategy) {struct s} : eProp :=
match s with
— (sLeaf po) ⇒ eTrue
— (sNode a c sl sr)
    ⇒ (eBI sl) && (eBI sr)
      && match c with
          — lchoice ⇒ eLeq ((stratPO sr) a)
                           ((stratPO sl) a)
          — rchoice ⇒ eLeq ((stratPO sl) a)
                           ((stratPO sr) a)
          end
end.
```

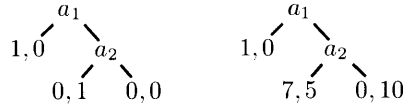
The defined function takes a strategy and, without fail, produces a propositional conjunction saying whether the strategy is a backwards induction equilibrium point.

Proposition 19 *eBI is a total, computable predicate (i.e., function into eProp).*

Proof By construction. □

In the *game* (i.e., *strategy* without *choices*) below on the left, there is one backwards induction equilibrium: a_1 and a_2 both chooses left and gets payoffs 1 and 0, respectively. The game has two Nash equilibria (i.e., strategies where no-one single-handedly can do better), as the overall outcome does not depend on a_2's choice. In the game on the right, only

a_1 choosing left and a_2 choosing right is either kind of equilibrium point. If a_2 chooses left, a_1 would go right but then a_2 would go back to the right, and a_1 would go back left. In other words, equilibria are not about universal optimality and the example nicely motivates the moniker "non-cooperative" to describe game theory based on Nash equilibria.



In [23], we prove by the same means used here that all sequential games have a backwards induction equilibrium and that these are all Nash equilibrium points, i.e., we Coq-verify an inductive proof of Kuhn's theorem [12].

4.5 Rationality Predicate

Aumann defines rationality informally as follows [2].

> *"Rationality of a player means ... that no matter where he finds himself — at which vertex — he will not knowingly continue with a strategy that yields him less than he could have gotten with a different strategy."*

In Aumann's formalism, the actual definition is $\bigcap_{v \in V_i} \bigcap_{t_i \in S_i} (\neg K_i[h_i^v(\mathbf{s}; t_i) > h_i^v(\mathbf{s})])$, [2, eq. (3)]. Stripping off the outermost intersection leaves us with having to consider the following big conjunction, where our p is Aumann's $h_i^v(\mathbf{s})$.

Coq-Formalism 20

```
Fixpoint Comp_nKns (a:G) (s:strategy) (p:payoff)
                   {struct s} : eProp :=
match s with
— (sLeaf po) ⇒ nKn a (eLeq (po a) p)
— (sNode a' c sl sr)
   ⇒ if (agentEq a a')
      then (Comp_nKns a sl p)
           && (Comp_nKns a sr p)
      else match c with
           — lchoice ⇒ (Comp_nKns a sl p)
           — rchoice ⇒ (Comp_nKns a sr p)
                   end
end.
```

Proposition 21 *Comp_nKns is a total, computable predicate.*

Proof By construction. □

The idea is that we recursively call the function along all paths that the agent, a, could (single-handedly) have decided to take inside the considered strategy, s: when we reach a sub-node owned by a, we pursue both options but when we reach a sub-node owned by another agent, we respect his choice. Any leaf we reach is used to create a conjunct saying that the agent does not know that the payoff he gets there is better than the one he originally decided he should go for. Full rationality is the big conjunction of *Comp_nKns*-results over all nodes owned by all agents, i.e., over all nodes in a strategy tree.

Coq-Formalism 22

```
Fixpoint eRat (s:strategy) {struct s} : eProp :=
match s with
— (sLeaf po) ⇒ eTrue
— (sNode a c sl sr)
   ⇒ (eRat sl) && (eRat sr)
        && (Comp_nKns a s ((stratPO s) a))
end.
```

Proposition 23 *eRat is a total, computable predicate.*

Proof By construction. □

This completes our formal(-ist) presentation of the framework that Aumann's theorem pertains to.

5 A First Formalist Analysis

As mentioned, Aumann's theorem states that it is the case for all strategies that if there is common knowledge that everybody behaves rationally in a given strategy, then that strategy is a backwards induction equilibrium point.

$$\forall\ s : strategy,\ (C\ (eRat\ s)) \Longrightarrow (eBI\ s)$$

5.1 An Example

Aumann's theorem is universally quantified over strategies. This means, for example, that the implication must hold for any strategy that is just a node with two leafs directly below it and the agent has chosen, say, the left branch.

Applying *eRat* to this example returns the following.

$$eTrue \;\&\&\; eTrue \;\&\&\; (nKn\ a\ (eLeq\ \mathrm{p_1}\ \mathrm{p_1})) \;\&\&\; (nKn\ a\ (eLeq\ \mathrm{p_2}\ \mathrm{p_1}))$$

Analogously, we get the following by applying *eBI*.

$$eTrue \;\&\&\; eTrue \;\&\&\; (eLeq\ \mathrm{p_2}\ \mathrm{p_1})$$

By the usual rules for conjunction, the occurrences of *eTrue* are superfluous and (our intention is that) *eLeq* $\mathrm{p_1}$ $\mathrm{p_1}$ will hold, too. In other words, Aumann's theorem mandates that the following implication must be provable.

$$C\ ((nKn\ a\ eTrue) \;\&\&\; (nKn\ a\ (eLeq\ p_2\ p_1))) \implies eLeq\ p_2\ p_1 \quad (1)$$

C distributes over conjunction and we are done if either $C\ (nKn\ a\ eTrue)$ is *eFalse*, in which case the theorem would be trivial because that conjunct is present for any node, or the following is provable.

$$C\ (nKn\ a\ (eLeq\ p_2\ p_1)) \implies eLeq\ p_2\ p_1 \quad (2)$$

As we have axiom T for C, see Lemma 8, axiom T for *nKn* would give us (2).

5.2 Subtleties of Various Axioms T

We have axiom T for K: (T_{K_a}), see Figure 2, including for negated properties.

$$K\ \mathrm{a}\ (Not\ p) \implies Not\ p \quad (3)$$

We recall that *nKn* is not-K-not and that not stands for "implies *eFalse*".

$$K\ \mathrm{a}\ (Not\ p) \implies p \implies eFalse$$

$$\frac{}{\Gamma \vdash^{\Delta} eDec(p) \supset \neg K_a(\neg p) \supset p}\ (dec\text{-}T_{nKn_a})$$

Fig. 7. Axiom decidable-T for *nKn*

As we may freely swap the order of left-hand sides of $\implies$, (T_{K_a}) thus implies:

$$p \implies nKn\ \mathrm{a}\ p$$

In other words, if we add axiom T for *nKn*, we collapse *nKn* because the axiom is nothing but the opposite of the preceding implication.

$$(T_{nKn_a}) \implies (p \iff nKn\ a\ p)$$

This is clearly not desirable in general as *nKn* is thought to have roughly the meaning of "to believe".[7] The question we are faced with in (2) is whether the C-modality qualifies *nKn* enough to accept axiom T for the combined modality. If we use the interpretation that *nKn* is belief, or even absence of doubt, it is difficult to see how common knowledge of that fact can impact on p but interpretations are, of course, of little technical relevance. Technically speaking, we have found no compelling proof-theoretic argument either for or against axiom T for C-compose-*nKn*.

We note, instead, that (2) holds trivially if it, in fact, is the case that *eLeq* $\mathrm{p_2}$ $\mathrm{p_1}$ holds. Conversely, *nKn a* (*eLeq* $\mathrm{p_2}$ $\mathrm{p_1}$) cannot be allowed to hold if *Not* (*eLeq* $\mathrm{p_2}$ $\mathrm{p_1}$) can be established independently because that would allow us to conclude that also *eLeq* $\mathrm{p_2}$ $\mathrm{p_1}$ holds, which would leave our formalism inconsistent. Consequently, we propose as a general principle that axiom T holds for *nKn* in case the property we are considering is decidable, i.e., if we definitely know whether it holds or not.

Definition 24 *Let eDec be a predicate expressing* decidability*; let axiom decidable-T for nKn be as defined in Figure 7.*

[7] Actually, *nKn* is slightly stronger than the usual belief modality, B, i.e., $nKn(p) \implies B(p)$. B is typically defined like K except without axiom T: $K(p) \Leftrightarrow B(p) \wedge p$. Our development also works with B instead of *nKn*.

We shall see in Section 6 that the resulting theory is, in fact, consistent. Informally, *the axiom asks agents to not believe in propositions that it is within their power to decide to be refutable.* This basically means that agents may not believe F.

5.3 The Inductive Insight

The example-based analysis above is complete in an interesting sense, as hinted to in Section 1.2. The issue is the exact details of the rationality predicate, see Coq-Formalism 22. For bigger strategy trees both *eRat* and *eBI* will produce bigger conjunctions than the ones considered above. However, *eRat*'s will grow in two dimensions while *eBI*'s will grow only in one. In the case of *eRat*, specifically, we will get more conjuncts both from *eRat* itself and from *Comp_nKns*. What Section 1.2 tells us is that most of the latter ones are superfluous. Indeed, what we call the "inductive insight" is that it is likely to (and actually does) suffice for the rationality predicate to have the same structure as the *eBI* predicate. We call this version *local rationality.*

Coq-Formalism 25

```
Fixpoint eLRat (s:strategy) {struct s} : eProp :=
match s with
— (sLeaf po) ⇒ eTrue
— (sNode a c sl sr)
    ⇒ (eLRat sl) && (eLRat sr)
        && match c with
          — lchoice ⇒ nKn a (eLeq ((stratPO sr) a)
                                  ((stratPO sl) a))
          — rchoice ⇒ nKn a (eLeq ((stratPO sl) a)
                                  ((stratPO sr) a))
          end
end.
```

Applying *eLRat* to the example at the beginning of this section gives.

$$eTrue \;\&\&\; eTrue \;\&\&\; (nKn\ a\ (eLeq\ p_2\ p_1))$$

More precisely, *eLRat* will always produce exactly one conjunct involving *nKn* for each node; it will be applied to the comparison of the induced payoffs in the left and right sub-strategies. In the case of *eRat*, *nKn* will be applied to the chosen payoff in any node compared with any conceivable alternative within the agent's reach, including the locally-determined one considered by *eLRat.*

$$\forall\ s : strategy,\ (eRat\ s \Longrightarrow eLRat\ s).$$

We will return to this point in Section 8.

6 (Constructive) Decidability

We note that the propositional part of our provability relation, see Figure 1, does not include *reductio ad absurdum*, below, but merely *ex falso quod libet*, ($E_{\mathbf{F}}$).

$$\frac{\Gamma, \neg p \vdash^{\Delta} \mathbf{F}}{\Gamma \vdash^{\Delta} p}\,(E_{\mathbf{F}}^{RAA})$$

This means that we are considering a constructive logic, specifically *intuitionistic logic*, rather than full *classical logic.* See Appendix A for an account of why the former is constructive while the latter is not.

All intuitionistically provable formulas are naturally also classically provable but not vice versa. Interestingly, though it is outside the scope of this article, we note that two mappings exist such that a formula is provable in one system if and only if the translated version is provable in the other system. While it therefore may seem like there is little difference between classical and intuitionistic logic, we shall take advantage of the constructive nature of the latter by noting that a constructively provable disjunction always has one of the disjuncts being provable. In other words, decidability can (and typically is) coded constructively as follows.

Coq-Formalism 26

```
Definition eDec (P:eProp) := P vv (Neg P).
```

The technical justification for this definition is given in Appendix A, Theorem 40. The advantage to us in using the stricter notion of intuitionistic provability is that decidability becomes simple to express and, sometimes, straightforward to prove. We are not aware of a similarly concise coding of

decidability either as a classical predicate or as a stand-alone modality. As it is, we can directly address our alternative version of Aumann's Theorem.

First, however, we note that another way of reading classical logic's *reductio ad absurdum* is that it mandates $\neg\neg p \supset p$, for all p, which is not guaranteed intuitionistically. That said, the other implications involving (at least) double-negation do hold in intuitionistic logic: $p \supset \neg\neg p$ and $\neg p \Leftrightarrow \neg\neg\neg p$. The point is this: double-negation does hold intuitionistically for decidable p.

Proposition 27 $\Gamma \vdash^{\Delta} p \vee (\neg p) \supset \neg\neg p \supset p$.

An interesting consequence is the following.

Lemma 28 *($dec\text{-}T_{nKn_a}$) is equivalent to*

$$\frac{}{\Gamma \vdash^{\Delta} \neg K_a(\neg p) \Leftrightarrow \neg\neg p}\ (\neg\neg\text{-}nKn_a)$$

Proof ($dec\text{-}T_{nKn_a}$) follows from ($\neg\neg\text{-}nKn_a$) according to Coq-Formalism 26 and Proposition 27. For the other direction, we first note that (3) implies $\neg\neg p \supset \neg K_a(\neg p)$ by contraposition. Secondly, $\neg K_a(\neg p) \supset \neg\neg p$ is itself equivalent to ($dec\text{-}T_{nKn_a}$) because $(p \vee \neg p) \supset p$ is equivalent to $\neg\neg p$ (by two reasonably direct arguments). □

In other words, adding axiom ($dec\text{-}T_{nKn_a}$) has the formal effect of mandating that nKn can only hold for propositions that are not explicitly refutable, which is the usual intuitionistic reading of double-negation. Moreover, adding the axiom is consistent with respect to the reading of nKn used by Aumann and with respect to intuitionistic logic. A consequence is that adding the axiom is logically consistent.

7 Decidable Local Rationality

In this section, we present two proofs that local rationality implies backwards induction in the presence of axiom ($dec\text{-}T_{nKn_a}$) and without reference to the common-knowledge modality. The first proof is general and abstract, merely asserting that payoff-comparison is decidable, while the second one shows what happens when we have an actual decision procedure at hand. Both proofs list the rather limited set of proof principles they use in addition to structural induction and computational reasoning, see Figures 4 and 5.

7.1 Abstract Version

In order to define a provability relation in Coq, we invoke Coq's *Prop*-sort, which is different from our *ePropp* but is constructed according to the same principles; implication in *Prop* is written →. The scheme we use is the one we accounted for in Section 2.7.

Coq-Formalism 29

```
Inductive eThm : eProp → Prop :=
— e_id : ∀ p : eProp, eThm (p ⟹ p)
— prj_33 : ∀ p1 p2 q1 q2 r1 r2 : eProp,
            (eThm (p1 ⟹ p2))
             → (eThm (q1 ⟹ q2))
             → (eThm (r1 ⟹ r2))
             → (eThm (p1 && q1 && r1
                      ⟹ p2 && q2 && r2))
— dec_T_nKn : ∀ a : G, ∀ p : eProp,
            eThm ((eDec p) ⟹ (nKn a p) ⟹ p)
— e_MP : ∀ p q : eProp,
            eThm (p ⟹ q) → eThm p → eThm q.

Notation "⊢ p" := (eThm p) (at level 85).
```

With this, we merely need to stipulate that our abstract pay-off ordering, see Coq-Formalism 13, is decidable and our inductive proof can proceed as described in Section 1.2.

Coq-Formalism 30

```
Axiom decOrd : ∀ po1 po2, ⊢ (eDec ((eLeq po1) po2)).

Theorem Dec_LRat_BI : ∀ s , ⊢ (eLRat s ⟹ eBI s).
Proof.
induction s.
  simpl. apply e_id.
  simpl.
  apply prj_33.
  apply IHs1.
  apply IHs2.
  induction c; (eapply e_MP ; [apply dec_T_nKn
                              | apply decOrd]).
Qed.
```

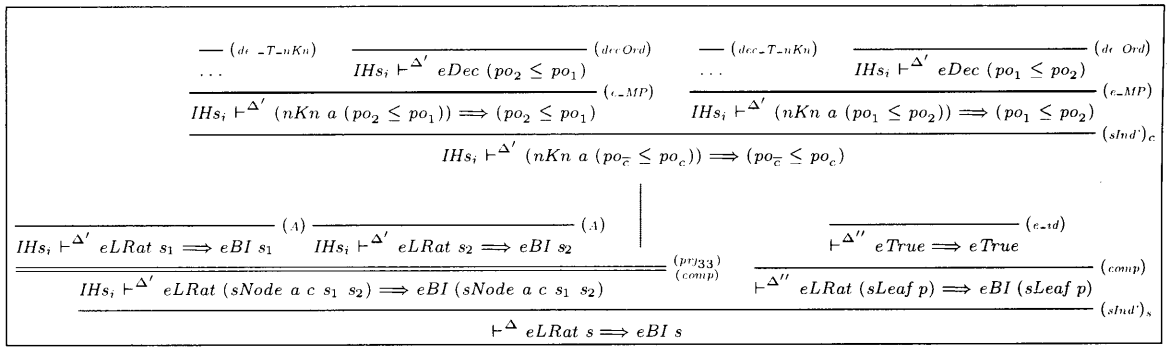


Fig. 8. Graphical proof of *Theorem Dec_LRat_BI* — detailed explanations in the text

Figure 8 contains a graphical presentation of the proof in the style of Section 2:

- Δ contains the Coq-Formalisms listed so far, as well as s of sort *strategy*;
- Δ' extends Δ with s_1, s_2 of sort *strategy*, a of sort G, and c of sort *choice*;
- Δ'' extends Δ with p of sort *payoffs*;
- $\bar{c}$ is short for the opposite choice of c;
- po_i is short for $(stratPO\ s_i\ a)$;
- proof rules (*comp*) and (*A*), aka (*Assm*), are borrowed from Coq via the tactics *apply*, *simpl*;
- the proof rule (*sInd'*) combines (*sInd*), cf. Figure 4, and ($I_{\supset}$), see Figure 1, and is borrowed from Coq via the tactic *induction*;
- IHs_i is short for IHs_1, IHs_2, which, in turn, is short for the induction hypotheses for *strategy*: $eLRat\ s_1 \Longrightarrow eBI\ s_1$ and $eLRat\ s_2 \Longrightarrow eBI\ s_2$;
- $(sInd')_c$ is structural induction over *choice*, which is degenerately inductive and, hence, the rule is not associated with induction hypotheses.

The proof starts by invoking structural induction on strategies, creating two cases for us to consider: one for internal nodes and one for leaves. The first line after invoking induction is for the leaf case. The command *simpl* indicates that we do computational reasoning, in order to unfold definitions, which leaves us with having to prove that *eTrue* implies *eTrue*. The node case starts again by defintion unfolding, which results in the three conjuncts from *eLRat*, i.e., (*eLRat s1*), (*eLRat s2*), and the considered use of *nKn* on the payoff-comparison, implying the three conjuncts from *eBI*, i.e., (*eBI s1*), (*eBI s2*), and the unqualified payoff-comparison. We then use our axiom *prj_33* in order to consider the pointwise implications between the conjuncts one by one. The first two follow by induction hypotheses, as discussed. For the last, we induct on the choice made in the node, followed by *dec_T_nKn* applied to *decOrd* with *e_MP*, aka ($E_{\supset}$).

7.2 Decision-Procedure Version

In this section, we consider the simple case of natural numbers as payoffs. We saw in Section 2.7 that we can give an ad hoc definition of the less-than-or-equal-to relation on natural numbers, which results in a rule-induction principle for proving the relationship. Alternatively, and because natural numbers themselves are inductively defined (and, thus, well-founded), see Section 2.5, we have a structural-recursion principle that we can use to actually decide whether the relationship holds or not. First, we introduce a sort for the answers of the function.

Coq-Formalism 31

```
Inductive eBool : U :=
— eTrue : eBool
— eFalse : eBool.
```

We then present our decision-procedure for less-than-or-equal-to on natural numbers.

Coq-Formalism 32

```
Fixpoint eLeqDP (n1 n2:Nat) {struct n1} : eBool :=
match n1,n2 with
— Zero , _ ⇒ eTrue
— Succ n1a, Zero ⇒ eFalse
— Succ n1a, Succ n2a ⇒ eLeqDP n1a n2a
end.
```

Proposition 33 *eLeqDP is a total, computable function.*

Proof By construction. □

With *eTrue* and *eFalse* defined separately, our new *eProp* imports them as propositional values that we definitely know what are and close them under the relevant logical connectives.

Coq-Formalism 34

```
Inductive eProp : Type :=
  — decprop : eBool → eProp
  — Imp : eProp → eProp → eProp
  — And : eProp → eProp → eProp
  — nKn : eProp → eProp.
```

Provability is defined basically as above, except that we no longer stipulate that axiom T for *nKn* can be applied to all decidable propositions. Instead, we require use of the sort *eBool*, which contains two constants and given one we will know which.

Coq-Formalism 35

```
Inductive eThm0 : eProp → Prop :=
  — dec_T : ∀ b : eBool,
        eThm0 (nKn (decprop b) ⟹ (decprop b))
  — e_id : ∀ p : eProp,
        eThm0 (p ⟹ p)
  — prj_33 : ∀ p1 p2 q1 q2 r1 r2 : eProp,
        (eThm0 (p1 ⟹ p2))
        → (eThm0 (q1 ⟹ q2))
        → (eThm0 (r1 ⟹ r2))
        → (eThm0 (p1 && q1 && r1
                  ⟹ p2 && q2 && r2)).
Notation "⊢0 p" := (eThm0 p) (at level 85).
```

With this, we can again prove that local rationality implies backwards induction.

Coq-Formalism 36

```
Theorem nat_LRat_BI : ∀ s, ⊢0 (eLRat s ⟹ eBI s).
Proof.
induction s.
  simpl. apply e_id.
  simpl.
  apply prj_33.
  apply IHs1.
  apply IHs2.
  induction c; apply dec_T.
Qed.
```

The slight simplification in the proof comes from the fact that we do not need to use modus ponens, due to the direct nature of axiom T for *nKn* on *eBool*s.

8 Aumann's Theorem

We now derive Aumann's original result by first proving that *eRat* implies *eLRat* using projections on conjunction. The needed proof principles are as follows.

Coq-Formalism 37

```
Inductive eThm' : eProp → Prop :=
— import : ∀ p : eProp,
          ⊢ p → eThm' p
— T_C : ∀ p : eProp,
          eThm' (C p ⟹ p)
— eId : ∀ p : eProp,
          eThm' (p ⟹ p)
— prj_33' : ∀ p1 p2 q1 q2 r1 r2 : eProp,
          (eThm' (p1 ⟹ p2))
            → (eThm' (q1 ⟹ q2))
            → (eThm' (r1 ⟹ r2))
            → (eThm' (p1 && q1 && r1
                      ⟹ p2 && q2 && r2))
— trans : ∀ p q r : eProp,
          (eThm' (p ⟹ q))
            → (eThm' (q ⟹ r))
```

$\rightarrow$ *(eThm' (p $\Longrightarrow$ r))*
— *prj_l : $\forall$ pl pr : eProp,*
eThm' (pl && pr $\Longrightarrow$ pl)
— *prj_r : $\forall$ pl pr : eProp,*
eThm' (pl && pr $\Longrightarrow$ pr).

Notation "$\vdash$' p" := (eThm' p) (at level 85).

Coq-Formalism 38
Lemma Rat_is_LRat : $\forall$ s , $\vdash$' (eRat s $\Longrightarrow$ eLRat s).

Please see Appendix B for the details.

Aumann's result can now be arrived at by composing the just-proved implication with our previously-established (and, thus, merely *import*ed) result that *eLRat* implies *eBI* followed by axiom T for *C*.

Coq-Formalism 39
Theorem Aumann_noC : $\forall$ s , $\vdash$' (eRat s $\Longrightarrow$ eBI s).
Proof.
intro.
eapply trans.
apply Rat_is_LRat.
apply import. apply Dec_LRat_BI.
Qed.

Theorem Aumann : $\forall$ s , $\vdash$' (C (eRat s) $\Longrightarrow$ eBI s).
Proof.
intro.
eapply trans.
apply T_C.
apply Aumann_noC.
Qed.

9 Conclusion

We have presented in detail a fully transparent proof of Aumann's Theorem on Rationality. The proof has been verified in the Coq proof assistant and the sources are available at the homepage of the first author, `http://www.jaist.ac.jp/~vester/`. Compared to existing mathematical approaches to the result, the main clarifying and simplifying contribution of our proof is the use of structural induction over strategies. We abandoned the seemingly established use of axiom T for the common-knowledge modality composed with the not-knowledge-not modality, due to inconclusive proof-theoretic justification for it. Instead, we introduced an axiom, $(dec\text{-}T_{nKn_a})$, *that asks agents to not believe in propositions that it is within their power to decide to be refutable.* The axiom is consistent with the fact that decidable propositions enjoy double-negation in intuitionistic logic. We hope our formal development will have also non-formal, e.g., philosophical or sociological, relevance to interested parties.

Acknowledgements We thank Michael Norrish for comments on the manuscript.

A Intuitionistic Logic

Intuitionistic logic enjoys the *disjunction property*, i.e., it is the case that $\varepsilon \vdash^{\Delta} p_l \vee p_r$ implies either $\varepsilon \vdash^{\Delta} p_l$ or $\varepsilon \vdash^{\Delta} p_r$ [7, 8]. Indeed, the following strong version of the disjunction property holds.

Theorem 40 (Feasible Disjunction [3]) *There is an algorithm that, given a proof of $\varepsilon \vdash^{\Delta} p_l \vee p_r$ according to the rules in Figure 1, produces a proof of $\varepsilon \vdash^{\Delta} p_l$ or a proof of $\varepsilon \vdash^{\Delta} p_r$ in polynomial time in the size of the original proof.*

The given complexity bound extends to include the modalities we use [6], see Figures 2 and 3, but the complexity becomes non-feasible when including function and relation symbols, see Figures 4–6, although an algorithm still exists [3]. Classical logic, by contrast, includes, e.g., *reductio ad absurdum*, $(E_{\mathbf{F}}^{RAA})$, in place of *ex falso quod libet*, $(E_{\mathbf{F}})$.

$$\frac{\Gamma, \neg p \vdash^{\Delta} \mathtt{F}}{\Gamma \vdash^{\Delta} p}\,(E_{\mathbf{F}}^{RAA})$$

Under $(E_{\mathbf{F}}^{RAA})$, $p_l \vee p_r$ becomes equivalent to $(\neg p_l) \supset p_r$, which implies that $p \vee \neg p$ is a tautology, for any p (aka *the law of excluded middle*). In particular, with $\Delta_t(x) = \textsc{eProp}$.

$$\cfrac{\cfrac{}{\neg x \vdash^{\Delta} \neg x}\,(Assm)}{\varepsilon \vdash^{\Delta} \neg x \supset \neg x}\,(I_{\rightarrow})$$

This means that we can prove classically that $x \vee \neg x$ is a theorem for any propositional variable but we cannot prove that either x or $\neg x$ is a theorem.

B Lemma *Rat_is_LRat*

Coq-Formalism 41

```
Lemma nKns_is_nKn : ∀ a p s,
  ⊢' ((Comp_nKns a s p)
        ⟹ nKn a (eLeq ((stratPO s) a) p)).
Proof.
induction s.
  simpl. apply eId'.
  simpl. induction agentEq.
  induction c.
  eapply trans. apply prj_l. apply IHs1.
  eapply trans. apply prj_r. apply IHs2.
  induction c. apply IHs1. apply IHs2.
Qed.

Lemma Rat_is_LRat : ∀ s , ⊢' (eRat s ⟹ eLRat s).
Proof.
induction s.
  simpl. apply eId.
  simpl. apply prj_33'.
  apply IHs1.
  apply IHs2.
  rewrite agentEqual. induction c.
  eapply trans. apply prj_r. apply nKns_is_nKn.
  eapply trans. apply prj_l. apply nKns_is_nKn.
Qed.
```

References

[1] Peter Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.7, pages 739–782. North-Holland, Amsterdam, 1977.

[2] Robert J. Aumann. Backward induction and common knowledge of rationality. *Games and Economic Behavior*, 8, 1995.

[3] Sam Buss and Grigori Mints. The complexity of the disjunction and existential properties in intuitionistic logic. *Annals of Pure and Applied Logic*, 99:93–104, 1999.

[4] Thierry Coquand and Gerard Huet. The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988.

[5] Gilles Dowek, Christine Paulin-Mohring, et al. Coq. http://coq.inria.fr/.

[6] Mauro Ferrari, Camillo Fiorentini, and Guido Fiorino. On the complexity of the disjunction property in intuitionistic and modal logics. *ACM Transactions on Computational Logic*, 6(3):519–538, 2005.

[7] Gerhard Gentzen. Untersuchungen über das logische Schliessen I, II. *Mathematische Zeitschrift*, 39:176–210,405–431, 1935. Translation appears pp. 68-131 in *The Collected Papers of Gerhard Gentzen*; North-Holland; Amsterdam. Edited and introduced by M.E. Szabo.

[8] Gerhard Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*. North-Holland, 1969.

[9] Joseph Y. Halpern. Substantive rationality and backward induction. *Games and Economic Behavior*, 37:425–435, 2001.

[10] Jaakko Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, New York, 1962.

[11] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 470–490. Academic Press, 1980.

[12] Harold W. Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games II*, 1953. Reprinted in [13].

[13] Harold W. Kuhn, editor. *Classics in Game Theory*. Princeton Uni. Press, 1997.

[14] Pierre Lescanne. Mechanizing epistemic logic with Coq. Research Report RR2004-27, LIP-ENS de Lyon, 2004.

[15] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36, 1950. Reprinted in [13].

[16] John F. Nash. *Non-Cooperative Games*. PhD thesis, Princeton University, 1950.

[17] Christine Paulin-Mohring. Inductive definitions in the system Coq: Rules and properties. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications, TLCA'93*, volume 664 of *Lecture Notes in Computer Science*, pages 328–345. Springer-Verlag, 1993.

[18] Dov Samet. Hypothetical knowledge and games with perfect information. *Games and Economic Behavior*, 17(2), 1996.

[19] Reinhard Selten. Spieltheoretische Behandlung eines Oligopolmodells mit Nachfrageträgheit. *Zeitschrift für die desamte Staatswissenschaft*, 121, 1965.

[20] Reinhard Selten. Reexamination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory*, 4, 1975. Reprinted in [13].

[21] Robert Stalnaker. Knowledge, belief and counterfactual reasoning in games. *Economics and Philosophy*, 12:133–162, 1996.

[22] The Coq Development Team. The Coq proof assistant reference manual, version 8.0. Technical report, INRIA, 2004. Available at [5].

[23] René Vestergaard. A constructive approach to sequential Nash equilibria. *Information Processing Letters*, 97:46–51, 2006.