When Do Types Induce the Same Belief Hierarchy? The Case of Finitely Many Types*

EPICENTER Working Paper No. 1 (2014)

Andrés Perea†
Maastricht University

First version: January 2014
This version: September 2014

Abstract

Harsanyi (1967–1968) showed how infinite belief hierarchies can be encoded by means of type structures. Such encodings, however, are far from unique: Two different types – possibly from two different type structures – may generate exactly the same belief hierarchy. In this paper we present a finite recursive procedure, the *Type Partitioning Procedure,* which verifies whether two types, from two potentially different finite type structures, induce the same belief hierarchy or not. Important is that the procedure does not make explicit reference to belief hierarchies, but operates entirely within the "language" of type structures. In the second part of this paper we relate the procedure to the notion of *type morphisms* and *hierarchy morphisms* between type structures.

JEL Classification: C72

Keywords: Epistemic game theory, types, belief hierarchies.

*This paper is a substantially revised version of an earlier paper with the same title. I would like to thank Pierpaolo Battigalli, Eddie Dekel, Amanda Friedenberg, Willemien Kets, Christian Nauerz, Miklós Pintér and Elias Tsakas for very useful comments on the earlier version.

†*Address*: EpiCenter and Department of Quantitative Economics, Maastricht University, P.O. Box 616, 6200 MD, Maastricht, The Netherlands.
E-mail: a.perea@maastrichtuniversity.nl
Web: http://www.epicenter.name/Perea/

1 Introduction

Belief hierarchies play a fundamental role in the modern analysis of games. In games with *incomplete information* – where players face uncertainty about the opponents' utilities – it is important to model what a player believes about his opponents' utility functions, what he believes about the opponents' beliefs about their opponents' utility functions, and so on (Harsanyi (1962, 1967–1968), Böge and Eisele (1979), Mertens and Zamir (1985), Ely and Pęski (2006), Dekel, Fudenberg and Morris (2007), Weinstein and Yildiz (2007) and others). But even in games with *complete information* – where the players' actual utility functions are transparent to everyone – belief hierarchies naturally enter the analysis when we investigate the belief a player has about his opponents' *choices,* the belief he has about the opponents' beliefs about their opponents' choices, and so on. Such belief hierarchies are the basis for many concepts in epistemic game theory, most of which build upon the central notion of *common belief in rationality* (Brandenburger and Dekel (1987), Tan and Werlang (1988)). For an overview of these concepts, see Brandenburger (2007), Perea (2012) and Dekel and Siniscalchi (2013).

An infinite belief hierarchy for a player starts with a *first-order* belief, which is a probability measure on the player's basic space of uncertainty. This basic space of uncertainty could contain the set of opponents' choices, the parameters that determine the players' utility functions (as in Harsanyi (1962, 1967–1968)), or even a combination of both (as in Böge and Eisele (1979) and Mertens and Zamir (1985)). A *second-order* belief for a player is a probability measure on both the basic space of uncertainty *and* the opponents' possible first-order beliefs. In particular, a second-order belief induces a belief about the opponents' first-order beliefs. In the same way, we can define third-order beliefs and higher. An infinite belief hierarchy consists of a first-order belief, a second-order belief, and so on, *ad infinitum.*

To the best of my knowledge, Harsanyi (1962) was the first to formally define a belief hierarchy within the context of a game, although he did so for a very special setting. One important practical problem with belief hierarchies – and that may also have been a reason why belief hierarchies entered the game theory stage relatively late – is that these are *infinite* objects, with *infinitely* many layers of beliefs. It is thus impossible to explicitly write down a belief hierarchy, layer by layer, as there are infinitely many of these. But then, the question naturally arises: Is there a way to represent belief hierarchies in a compact and convenient way?

Harsanyi (1967–1968), some years after he introduced the notion of a belief hierarchy, gave a positive and elegant answer to this question. He focused on a setting in which the belief hierarchies concern only the players' *utilities,* but his construction has later been extended to situations where players also hold beliefs about the opponents' *choices.* The construction that Harsanyi proposed was very simple: For every player we define a set of types, and for every type we define a utility function, together with a probabilistic belief about the *opponents' types.* From this very simple construction we can then derive, for every type, a first-order belief about the opponents' utility functions, a second-order belief about the opponents' first-order beliefs, and so on. That is, for every type we can derive a *full belief hierarchy* on the players' possible

utility functions in the game. This construction by Harsanyi was a major step forward, as it allowed us to encode infinite belief hierarchies about utilities in a very compact and convenient fashion.

Harsanyi's original idea can easily be adapted to a framework where players also hold beliefs about other features besides the players' utilities. Assume that every player faces a basic space of uncertainty, which can include the parameters determining the players' utility functions, the set of opponents' choices, and possibly some other features as well. Now, consider for every player a set of types, and associate to every type a probabilistic belief about the basic space of uncertainty *and* the opponents' types. Then, similarly to Harsanyi's construction, we can derive for every type a full belief hierarchy, specifying a first-order belief about the basic space of uncertainty, a second-order belief about the opponents' first-order beliefs about *their* basic spaces of uncertainty, and so on. This construction, which we call a *type structure*, thus allows us to encode infinite belief hierarchies about any set of parameters in a very compact way.

Such encodings, however, are far from unique: Two different types – possibly from two different type structures – may encode one and the same belief hierarchy. In view of this "multiplicity problem" we ask the following natural question in this paper: When do two types, from two potentially different finite type structures, induce the same belief hierarchy?

Checking this directly, by explicitly comparing their induced first-order beliefs, second-order beliefs, and so on, may be quite cumbersome as one needs to check for infinitely many levels of belief. Instead, this paper presents a finite recursive procedure, the *Type Partitioning Procedure,* which tells us precisely when two such types induce the same belief hierarchy, and when they do not. The procedure works as follows. If we compare two types from two *different* type structures, we start by merging the two type structures into one large type structure. In every round, the procedure generates for every player a *partition* of the set of types in the large type structure, where the partition in the current round will always be a *refinement* of the partition from the previous round. Since we restrict to type structures with finitely many types, this procedure will always terminate within finitely many rounds. The equivalence classes in the final partitions will then be exactly those groups of types that generate the same belief hierarchy. That is, two types generate the same belief hierarchy exactly when they belong to the same equivalence class in the final partition. We actually show a bit more in Theorem 2: We prove that for every n, two types share the same n-th order belief precisely when they belong to the same equivalence class of the partition produced in round n of the procedure. In that sense, the *Type Partitioning Procedure* provides a convenient and automated way to verify whether two types share the same belief hierarchy, or the same beliefs up to a fixed order n. Important, moreover, is that the *Type Partitioning Procedure* does not make any explicit reference to belief hierarchies – it operates entirely within the "language" of type structures.

With Theorem 2 at hand, we can use the *Type Partitioning Procedure* to examine certain properties of individual type structures, or to investigate relations *between* type structures. The procedure can be used, for instance, to verify whether a given type structure is *redundant,* in the sense that it contains several different types inducing the same belief hierarchy. Or we can

apply it to check whether two different type structures generate the same collection of belief hierarchies, or whether one type structure is "smaller" than the other, in the sense that the collection of belief hierarchies of the first structure is contained in that of the second. This is the type of question investigated in, for instance, Friedenberg and Meier (2011).

Testing for such properties is important for the analysis of games with complete and incomplete information. Consider, for instance, Dekel, Fudenberg and Morris (2007) who show that in an incomplete information setting, two types generating the same belief hierarchy about the players' utilities will always induce the same set of *interim correlated rationalizable* choices[1]. Hence, if we want to check whether two different types – or two different type structures – generate the same sets of *interim correlated rationalizable* choices, it is sufficient to verify that they induce the same belief hierarchy – or collection of belief hierarchies – about the players' utilities, and this can be done through the *Type Partitioning Procedure.* Or consider Liu (2009) who shows that every redundant type structure in an incomplete information setting can be transformed into a non-redundant type structure in such a way that the set of Bayesian Nash equilibria is preserved. To see whether such a transformation is necessary or not one first has to check for redundancies in the type structure at hand, and also this can be done by running the *Type Partitioning Procedure.*

In this paper we also use the *Type Partitioning Procedure* to establish an interesting property of belief hierarchies which – I believe – is new. Suppose we compare two types – possibly from two different type structures – and let N be the total number of types in these two type structures. Then, we prove in Corollary 1 that these two types induce the same belief hierarchy exactly when they induce the same N-th order belief. To prove this result we use the property that the *Type Partitioning Procedure* will always terminate within at most N rounds. In particular, the smaller the number of types in the type structure, the less belief levels we must check in order to conclude that two types share the same belief hierarchy.

In the second part of the paper we link the partitions generated by the *Type Partitioning Procedure* to the notions of *type morphisms* and *hierarchy morphisms* between two type structures. A *type morphism* is a function that assigns to every type from the first type structure a "placeholder" in the second type structure, such that the types' beliefs about the opponents' choice-type combinations are "invariant" under this assignment. In particular, a type morphism makes no explicit reference to belief hierarchies. A *hierarchy morphism,* on the other hand, *does* explicitly refer to belief hierarchies, as it is defined as function assigning to every type from the first type structure a type from the second type structure that induces the same belief hierarchy.

It is well-known that every type morphism is a hierachy morphism (Heifetz and Samet, 1998) but not *vice versa* (Friedenberg and Meier, 2011). In fact, Friedenberg and Meier (2011) show, by means of an example, that for two types sharing the same belief hierarchy it may not be

[1]Ely and Pęski (2006) show that this is not true for *interim independent rationalizability* and *Bayesian Nash equilibrium*: Two types generating the same belief hierarchy about the players' utilities may generate different interim independent rationalizable choices, and different Bayesian Nash equilibrium choices.

possible to find a type morphism that maps the first type to the second. In that sense, type morphisms are "too restrictive" to characterize types that induce the same belief hierarchy.

We show that the final partitions of the *Type Partitioning Procedure* naturally induce a generalized version of a type morphism – which we call *set-valued type morphism.* The crucial difference with "traditional" type morphisms is that a set-valued type morphism maps a type from the first structure to a *set of types* from the second structure – rather than a single type – and this set may be empty. Set-valued type morphisms reduce to "traditional" type morphisms, however, if we require this set of types always to be a singleton. In Theorem 4 we show how set-valued type morphisms can be used to characterize those types that share the same belief hierarchy: Two types t and t' – possibly from different type structures – induce the same belief hierarchy precisely when there is a set-valued type morphism from the first type structure to the second type structure, mapping t to a set of types that contains t'. For the proof of this result we heavily rely on Theorem 2 above. Important is that set-valued type morphisms, like "traditional" type morphisms, do not make explicit reference to belief hierarchies.

In the last part of Section 4 we finally relate the partitions generated by the *Type Partitioning Procedure* to the notion of *hierarchy morphisms,* as studied in Friedenberg and Meier (2011). In Corollary 3 we use the *Type Partitioning Procedure* to give an easy characterization of hierarchy morphisms, relying substantially on Theorem 2. Since every type morphism is a hierarchy morphism but not the other way around – see our discussion above – a natural question is under which circumstances the two notions coincide. This is a question intensively studied in Friedenberg and Meier (2011), who show that the two notions are equivalent if the second type structure is non-redundant – at least when we stick to finite type structures as we do in this paper. In Corollary 4 we show how this result can be derived entirely from the insights in this paper.

The outline of this paper is as follows. In Section 2 we introduce type structures and belief hierarchies, and show how we can derive belief hierarchies from types. In Section 3 we describe the *Type Partitioning Procedure,* illustrate it by means of an example, and show how it characterizes those types that share the same belief hierarchy. In Section 4 we relate the partitions induced by the *Type Partitioning Procedure* to type morphisms and hierarchy morphisms. In Section 5 we discuss some possible extensions of this work. Section 6, finally, contains all proofs.

2 Belief Hierarchies and Types

In this section we show how belief hierarchies can be encoded by means of a type structure, and how every type within a type structure can be "decoded" by deriving a full belief hierarchy from it.

2.1 Encoding Belief Hierarchies by Type Structures

Consider a finite set of agents I. Assume that each agent i faces a *basic space of uncertainty* (X_i, Σ_i), where X_i is an arbitrary set and Σ_i a σ-algebra on X_i. That is, (X_i, Σ_i) is a measurable space. The combination $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$ of basic uncertainty spaces is called a *multi-agent uncertainty space.*

If the agents are the players in a game, the basic space of uncertainty for player i could, for instance, be the set of opponents' choice combinations, or the set of parameters determining the utility functions of the players, or even a combination of the two. The first scenario is the standard framework for games with complete information, the second scenario is Harsanyi's (1967–1968) original setting for games with incomplete information, whereas the last scenario is investigated in Böge and Eisele (1979) and Mertens and Zamir (1985), among others. The sets X_i could also include external events that cannot be influenced by the agents, as is the case in Böge and Eisele (1979).

A *belief hierarchy* for player i specifies a probability measure on $\mathcal{X}_i$ – the *first-order* belief, a probability measure on $\mathcal{X}_i$ and the opponents' possible first-order beliefs – the *second-order* belief, and so on. Following Harsanyi's (1967–1968) approach, we will encode such infinite belief hierarchies by means of *type structures.* In this paper we focus on type structures with *finitely* many types, which of course imposes restrictions on the possible belief hierarchies we can encode. Indeed, there are belief hierarchies which can simply not be encoded by type structures with finitely many types. See Section 5 for a discussion of this restriction to finite type spaces.

Definition 1 (Type Structure) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$. A **finite type structure** for $\mathcal{X}$ is a tuple $\mathcal{T} = (T_i, b_i)_{i \in I}$ where, for every player i,*

(a) T_i is the finite set of types for player i, and

(b) $b_i : T_i \to \Delta(X_i \times T_{-i}, \hat{\Sigma}_i)$ is a mapping that assigns to every type t_i a probabilistic belief $b_i(t_i) \in \Delta(X_i \times T_{-i}, \hat{\Sigma}_i)$ on his basic uncertainty space and the opponents' type combinations.

Here, $T_{-i} := \times_{j \neq i} T_j$. For any measurable space $(Y_i, \hat{\Sigma}_i)$, we denote by $\Delta(Y_i, \hat{\Sigma}_i)$ the set of probability measures on $(Y_i, \hat{\Sigma}_i)$. In part (b) of the definition, we assume $\hat{\Sigma}_i$ to be the product σ-algebra on $X_i \times T_{-i}$ induced by the σ-algebra Σ_i on X_i and the discrete σ-algebra on the finite set T_{-i}.

2.2 From Type Structures to Belief Hierarchies

In the previous subsection we have introduced a type structure as a way to encode belief hierarchies. We will now show how to "decode" a type within a type structure, by deriving the full belief hierarchy it induces.

Consider a finite type structure $\mathcal{T} = (T_i, b_i)_{i \in I}$ for Γ. Then, every type t_i within $\mathcal{T}$ induces an infinite belief hierarchy

$$h_i(t_i) = (h_i^1(t_i), h_i^2(t_i), ...),$$

where $h_i^1(t_i)$ is the induced first-order belief, $h_i^2(t_i)$ is the induced second-order belief, and so on. We will inductively define, for every n, the n-th order beliefs induced by types t_i in $\mathcal{T}$, building upon the $(n-1)$-th order beliefs that have been defined in the preceding step. We start by defining the first-order beliefs.

For every player i, and every type $t_i \in T_i$, define the first-order belief $h_i^1(t_i) \in \Delta(X_i, \Sigma_i)$ by

$$h_i^1(t_i)(E_i) := b_i(t_i)(E_i \times T_{-i}) \text{ for all } E_i \in \Sigma_i.$$

Now, suppose that $n \geq 2$, and assume that the $(n-1)$-th order beliefs $h_i^{n-1}(t_i)$ have been defined for all players i, and every type $t_i \in T_i$. Let

$$h_i^{n-1}(T_i) := \{h_i^{n-1}(t_i) \mid t_i \in T_i\}$$

be the finite set of $(n-1)$-th order beliefs for player i induced by types in T_i. For every $h_i^{n-1} \in h_i^{n-1}(T_i)$, let

$$T_i[h_i^{n-1}] := \{t_i \in T_i \mid h_i^{n-1}(t_i) = h_i^{n-1}\}$$

be the set of types in T_i that have the $(n-1)$-th order belief h_i^{n-1}.

Let $h_{-i}^{n-1}(T_{-i}) := \times_{j \neq i} h_j^{n-1}(T_j)$, and for a given $h_{-i}^{n-1} = (h_j^{n-1})_{j \neq i}$ in $h_{-i}^{n-1}(T_{-i})$ let $T_{-i}[h_{-i}^{n-1}] := \times_{j \neq i} T_j[h_j^{n-1}]$.

We define the n-th order beliefs $h_i^n(t_i)$ as follows. Let Σ_i^{n-1} be the product σ-algebra on $X_i \times h_{-i}^{n-1}(T_{-i})$ induced by the σ-algebra Σ_i on X_i and the discrete σ-algebra on the finite set $h_{-i}^{n-1}(T_{-i})$. For every type $t_i \in T_i$, let the n-th order belief $h_i^n(t_i) \in \Delta(X_i \times h_{-i}^{n-1}(T_{-i}), \Sigma_i^{n-1})$ be given by

$$h_i^n(t_i)(E_i \times \{h_{-i}^{n-1}\}) := b_i(t_i)(E_i \times T_{-i}[h_{-i}^{n-1}]) \tag{1}$$

for every $E_i \in \Sigma_i$ and every $h_{-i}^{n-1} \in h_{-i}^{n-1}(T_{-i})$.

Finally, for every type $t_i \in T_i$, we denote by

$$h_i(t_i) := (h_i^n(t_i))_{n \in \mathbb{N}}$$

the *belief hierarchy* induced by t_i.

The way we define belief hierarchies differs in one important way from, for instance, Heifetz and Samet (1998) and Friedenberg and Meier (2011). In the latter two works, the n-th order belief $h_i^n(t_i)$ is assumed to explicitly incorporate the $(n-1)$-th order belief $h_i^{n-1}(t_i)$ as a component, whereas this is not the case in our setting. See equation (1) above. Because of this, two types that share the same n-th order belief in the Heifetz-Samet and Friedenberg-Meier setting, automatically will share the same $(n-1)$-th order belief. This is not so obvious in our case. However, we will *prove,* in Corollary 1, that this property holds for our setting as well. That is, even though we do not explicitly build in the $(n-1)$-th order belief into the n-th order belief, we show that the $(n-1)$-th order belief can be *derived* from the n-th order belief. Our belief hierarchies are therefore "shorter" than those considered in Heifetz and Samet (1998) and Friedenberg and Meier (2011), while containing exactly the same amount of information.

3 Type Partitioning Procedure

Suppose we consider a finite type structure for some multi-agent uncertainty space $\mathcal{X}$. In this section we will present a recursive procedure – the *Type Partitioning Procedure* – that tells us, within finitely many steps, which types from this type structure induce the same belief hierarchy and which do not. Important is that this procedure does not make explicit reference to belief hierarchies, but operates entirely within the "language" of type structures.

3.1 Definition of the Procedure

To formally define this procedure, we need the following terminology. A *finite partition* of a set A is a finite collection $\mathcal{P} = \{P_1, ..., P_K\}$ of nonempty subsets $P_k \subseteq A$ such that $\cup_{k=1}^{K} P_k = A$ and $P_k \cap P_m = \emptyset$ whenever $k \neq m$. We refer to the sets P_k as *equivalence classes.* For an element $a \in A$, we denote by $\mathcal{P}(a)$ the equivalence class P_k to which a belongs. The *trivial partition* of A is the partition $\mathcal{P} = \{A\}$ containing only one set – the full set A. For two partitions $\mathcal{P}^1$ and $\mathcal{P}^2$ on A, we say that $\mathcal{P}^1$ is a *refinement* of $\mathcal{P}^2$ if for every set $P^1 \in \mathcal{P}^1$ there is a set $P^2 \in \mathcal{P}^2$ such that $P^1 \subseteq P^2$. We say that $\mathcal{P}^1$ is a *strict refinement* of $\mathcal{P}^2$ if $\mathcal{P}^1$ is a *refinement* of $\mathcal{P}^2$ and $\mathcal{P}^1 \neq \mathcal{P}^2$.

In the procedure we recursively partition the set of types of an agent into equivalence classes – starting from the trivial partition, and refining the previous partition with every step – until these partitions cannot be refined any further. We show that the equivalence classes produced in round n contain exactly the types that induce the same belief hierarchy up to order n. In particular, the equivalence classes produced at the end contain precisely those types that induce the same (infinite) belief hierarchy.

Procedure 1 (Type Partitioning Procedure) *Consider a multi-agent uncertainty space* $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, *and a finite type structure* $\mathcal{T} = (T_i, b_i)_{i \in I}$ *for* $\mathcal{X}$.

Initial step. *For every agent* i, *let* $\mathcal{P}_i^0$ *be the trivial partition of his set of types* T_i.

Inductive step. *Suppose that* $n \geq 1$, *and that the partitions* $\mathcal{P}_i^{n-1}$ *have been defined for every agent* i. *Then, for every agent* i, *and every* $t_i \in T_i$,

$$\mathcal{P}_i^n(t_i) = \{t_i' \in T_i \mid b_i(t_i')(E_i \times P_{-i}^{n-1}) = b_i(t_i)(E_i \times P_{-i}^{n-1}) \quad \text{for all } E_i \in \Sigma_i, \text{ and all } P_{-i}^{n-1} \in \mathcal{P}_{-i}^{n-1}\}. \tag{2}$$

The procedure ***terminates at round*** n *whenever* $\mathcal{P}_i^n = \mathcal{P}_i^{n-1}$ *for every agent* i.

In this procedure, $\mathcal{P}_{-i}^{n-1}$ is the partition of the set T_{-i} induced by the partitions $\mathcal{P}_j^{n-1}$ on T_j. More precisely, if $t_{-i} = (t_j)_{j \neq i}$ is in T_{-i}, then

$$\mathcal{P}_{-i}^{n-1}(t_{-i}) := \times_{j \neq i} \mathcal{P}_j^{n-1}(t_j),$$

Type structure $\mathcal{T} = (T_1, T_2, b_1, b_2)$
$T_1 = \{t_1, t_1', t_1''\}, \qquad T_2 = \{t_2, t_2', t_2''\}$
$b_1(t_1) = \frac{1}{2}(c, t_2) + \frac{1}{2}(d, t_2')$ $b_1(t_1') = \frac{1}{6}(c, t_2) + \frac{1}{3}(c, t_2'') + \frac{1}{2}(d, t_2')$ $b_1(t_1'') = \frac{1}{2}(c, t_2') + \frac{1}{2}(d, t_2'')$
$b_2(t_2) = \frac{1}{4}(e, t_1) + \frac{1}{2}(e, t_1') + \frac{1}{4}(f, t_1'')$ $b_2(t_2') = \frac{1}{8}(e, t_1) + \frac{1}{8}(e, t_1') + \frac{3}{4}(f, t_1'')$ $b_2(t_2'') = \frac{3}{8}(e, t_1) + \frac{3}{8}(e, t_1') + \frac{1}{4}(f, t_1'')$

Table 1: The type structure from Example 1

which is a subset of T_{-i}.

We will now illustrate the *Type Partitioning Procedure* by means of an example.

Example 1. Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$ where $I = \{1, 2\}$, $X_1 = \{c, d\}, X_2 = \{e, f\}$, and Σ_1, Σ_2 are the discrete σ-algebras on X_1 and X_2, respectively. Consider the type structure $\mathcal{T} = (T_1, T_2, b_1, b_2)$ in Table 1. Here, $b_1(t_1) = \frac{1}{2}(c, t_2) + \frac{1}{2}(d, t_2')$ means that type t_1 assigns probability $\frac{1}{2}$ to the pair $(c, t_2) \in X_1 \times T_2$, and probability $\frac{1}{2}$ to the pair $(d, t_2') \in X_1 \times T_2$. Similarly for the other types in the table. We will now run the *Type Partitioning Procedure.*

Initial Step. Let $\mathcal{P}_1^0$ be the trivial partition of the set of types T_1, and let $\mathcal{P}_2^0$ be the trivial partition of the set of types T_2. That is,

$$\mathcal{P}_1^0 = \{\{t_1, t_1', t_1''\}\} \text{ and } \mathcal{P}_2^0 = \{\{t_2, t_2', t_2''\}\}.$$

Round 1. By equation (2),

$$\begin{gathered}\mathcal{P}_1^1(t_1) = \{\tau_1 \in T_1 \mid \\ b_1(\tau_1)(\{c\} \times T_2) = b_1(t_1)(\{c\} \times T_2) = \tfrac{1}{2}, \\ b_1(\tau_1)(\{d\} \times T_2) = b_1(t_1)(\{d\} \times T_2) = \tfrac{1}{2}\} \\ = \{t_1, t_1', t_1''\},\end{gathered}$$

which implies that

$$\mathcal{P}_1^1 = \mathcal{P}_1^0 = \{\{t_1, t_1', t_1''\}\}.$$

At the same time,

$$\begin{gathered}\mathcal{P}_2^1(t_2) = \{\tau_2 \in T_2 \mid \\ b_2(\tau_2)(\{e\} \times T_1) = b_2(t_2)(\{e\} \times T_1) = \tfrac{3}{4}, \\ b_2(\tau_2)(\{f\} \times T_1) = b_2(t_2)(\{f\} \times T_1) = \tfrac{1}{4}\} \\ = \{t_2, t_2''\}\end{gathered}$$

which implies that $\mathcal{P}_2^1(t_2') = \{t_2'\}$, and hence

$$\mathcal{P}_2^1 = \{\{t_2, t_2''\}, \{t_2'\}.$$

Round 2. By equation (2),

$$\begin{gathered}\mathcal{P}_1^2(t_1) = \{\tau_1 \in T_1 \mid \\ b_1(\tau_1)(\{c\} \times \{t_2, t_2''\}) = b_1(t_1)(\{c\} \times \{t_2, t_2''\}) = \tfrac{1}{2}, \\ b_1(\tau_1)(\{c\} \times \{t_2'\}) = b_1(t_1)(\{c\} \times \{t_2'\}) = 0, \\ b_1(\tau_1)(\{d\} \times \{t_2, t_2''\}) = b_1(t_1)(\{d\} \times \{t_2, t_2''\}) = 0, \\ b_1(\tau_1)(\{d\} \times \{t_2'\}) = b_1(t_1)(\{d\} \times \{t_2'\}) = \tfrac{1}{2}\} \\ = \{t_1, t_1'\},\end{gathered}$$

which implies that $\mathcal{P}_1^2(t_1'') = \{t_1''\}$, and hence

$$\mathcal{P}_1^2 = \{\{t_1, t_1'\}, \{t_1''\}\}.$$

Since $\mathcal{P}_1^1 = \mathcal{P}_1^0$, we may immediately conclude that

$$\mathcal{P}_2^2 = \mathcal{P}_2^1 = \{\{t_2, t_2''\}, \{t_2'\}\}.$$

Round 3. As $\mathcal{P}_2^2 = \mathcal{P}_2^1$, we may immediately conclude that

$$\mathcal{P}_1^3 = \mathcal{P}_1^2 = \{\{t_1, t_1'\}, \{t_1''\}\}.$$

By equation (2),

$$\begin{gathered}\mathcal{P}_2^3(t_2) = \{\tau_2 \in T_2 \mid \\ b_2(\tau_2)(\{e\} \times \{t_1, t_1'\}) = b_2(t_2)(\{e\} \times \{t_1, t_1'\}) = \tfrac{3}{4}, \\ b_2(\tau_2)(\{e\} \times \{t_1''\}) = b_2(t_2)(\{e\} \times \{t_1''\}) = 0, \\ b_2(\tau_2)(\{f\} \times \{t_1, t_1'\}) = b_2(t_2)(\{f\} \times \{t_1, t_1'\}) = 0, \\ b_2(\tau_2)(\{f\} \times \{t_1''\}) = b_2(t_2)(\{f\} \times \{t_1''\}) = \tfrac{1}{4}\} \\ = \{t_2, t_2''\},\end{gathered}$$

which implies that $\mathcal{P}_2^3(t_2') = \{t_2'\}$, and hence

$$\mathcal{P}_2^3 = \{\{t_2, t_2''\}, \{t_2'\}\} = \mathcal{P}_2^2.$$

As $\mathcal{P}_1^3 = \mathcal{P}_1^2$ and $\mathcal{P}_2^3 = \mathcal{P}_2^2$, the procedure terminates at round 3. The final partitions of the types are thus given by

$$\mathcal{P}_1^\infty = \{\{t_1, t_1'\}, \{t_1''\}\} \text{ and } \mathcal{P}_2^\infty = \{\{t_2, t_2''\}, \{t_2'\}\}.$$

The reader may check that all types within the same equivalence class indeed induce the same belief hierarchy. That is, t_1 induces the same belief hierarchy as t_1', and t_2 induces the same belief hierarchy as t_2''. Moreover, t_1 and t_1'' induce different belief hierachies, and so do t_2 and t_2'. □

3.2 Characterization Result

We will now show that the *Type Partitioning Procedure* identifies precisely those types that share the same belief hierarchy. As a preparatory result, we will first highlight two important properties of the procedure. The first property states that the procedure is *monotonic* in the sense that the partitions generated at a particular round will always be *refinements* of the partitions generated in the round before. The second property states that the number of rounds that is needed for the procedure to terminate can never be larger than the total number of types we consider.

Theorem 1 (Properties of the Procedure) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, and a finite type structure $\mathcal{T} = (T_i, b_i)_{i \in I}$ for $\mathcal{X}$. For every agent i and every round $n \geq 0$, let $\mathcal{P}_i^n$ be the partition of T_i generated in round n of the Type Partitioning Procedure. Let N be the total number of types in $\cup_{i \in I} T_i$. Then,*

(a) the partition $\mathcal{P}_i^n$ will always be a refinement of $\mathcal{P}_i^{n-1}$, for all agents i and all $n \geq 1$;

(b) the procedure will terminate after at most N rounds.

With this result at hand we can now prove the main theorem in this paper, which states that the *Type Partitioning Procedure* characterizes precisely those groups of types that induce the same belief hierarchy. We actually prove a little more: we show that the partitions generated in round n of the procedure characterize exactly those types that yield the same n-th order belief.

Theorem 2 (Characterization Result) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, and a finite type structure $\mathcal{T} = (T_i, b_i)_{i \in I}$ for $\mathcal{X}$. For every agent i and every round $n \geq 0$, let $\mathcal{P}_i^n$ be the partition of T_i generated in round n of the Type Partitioning Procedure. Let $\mathcal{P}_i^\infty$ be the final partition generated by the procedure. Then, for every agent i, every $n \geq 1$, and every*

two types $t_i, t_i' \in T_i$, we have that

(a) $h_i^n(t_i) = h_i^n(t_i')$, if and only if, $t_i' \in \mathcal{P}_i^n(t_i)$;

(b) $h_i(t_i) = h_i(t_i')$, if and only if, $t_i' \in \mathcal{P}_i^\infty(t_i)$.

By combining Theorems 1 and 2 we can derive some interesting facts about finite type structures and belief hierarchies, which we state in the following corollary.

Corollary 1 (Properties of Belief Hierarchies) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, and a finite type structure $\mathcal{T} = (T_i, b_i)_{i \in I}$ for $\mathcal{X}$. Let N be the total number of types in $\cup_{i \in I} T_i$, and let $t_i, t_i' \in T_i$. Then,*

(a) for every $n \geq 2$, $h_i^{n-1}(t_i) = h_i^{n-1}(t_i')$ whenever $h_i^n(t_i) = h_i^n(t_i')$;

(b) $h_i(t_i) = h_i(t_i')$, if and only if, $h_i^N(t_i) = h_i^N(t_i')$.

Property (a) thus states that two types agreeing on the n-th order belief will also agree on all lower order beliefs. That is, the n-th order belief completely determines the first-order belief, the second-order belief, until the $(n-1)$-th order belief. Property (b) says that in order to check whether two types share the same infinite belief hierarchy or not we only have to compare the N-th order beliefs, where N is the total number of types in the type structure. To the best of my knowledge, this result is new in the literature.

The proof of this corollary is actually very easy. To show property (a) consider two types t_i, t_i' with $h_i^n(t_i) = h_i^n(t_i')$. Then, by Theorem 2, $t_i' \in \mathcal{P}_i^n(t_i)$. Since, by Theorem 1, $\mathcal{P}_i^n$ is a refinement of $\mathcal{P}_i^{n-1}$, it follows that $t_i' \in \mathcal{P}_i^{n-1}(t_i)$ and hence, by Theorem 2, $h_i^{n-1}(t_i) = h_i^{n-1}(t_i')$.

To show property (b), take two types t_i, t_i' with $h_i^N(t_i) = h_i^N(t_i')$. Then, by Theorem 2, $t_i' \in \mathcal{P}_i^N(t_i)$. By Theorem 1 we know that the procedure terminates after at most N round, and hence $\mathcal{P}_i^N = \mathcal{P}_i^\infty$. By Theorem 2 we conclude that $h_i(t_i) = h_i(t_i')$.

Some readers may ask why property (a) requires a proof, as in most other papers in the literature the n-th order belief induced by a type explicitly contains the $(n-1)$-th order belief as a component, and hence property (a) holds trivially. See, for instance, Heifetz and Samet (1998) and Friedenberg and Meier (2011). However, this is not the case in our setting: Our definition of $h_i^n(t_i)$ does not explicitly carry $h_i^{n-1}(t_i)$ as a component, and it is therefore not obvious that the n-th order belief fully determines the $(n-1)$-th order belief. This is a result, which requires a proof in our setting.

3.3 Comparing Types from Different Type Structures

We have seen that the *Type Partitioning Procedure* tells us exactly which types within a given type structure $\mathcal{T}$ induce the same belief hierarchy, and which do not. But what if we want to compare types from *different* type structures? We will see that the procedure will work for such settings as well.

Let us consider two different finite type structures, $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ and $\mathcal{T}^2 = (T_i^2, b_i^2)_{i \in I}$, for the same multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$. For a given agent i, take a type $t_i^1 \in T_i^1$ and a type $t_i^2 \in T_i^2$. How can we check whether t_i^1 and t_i^2 induce the same belief hierarchy?

What we can do is to first merge the two type structures into one large type structure, and to subsequently run the *Type Partitioning Procedure* for the large type structure. More precisely, let $\mathcal{T} = (T_i, b_i)_{i \in I}$ be the "large" type structure, where $T_i := T_i^1 \cup T_i^2$ for all agents i, and

$$b_i(t_i) := \begin{cases} b_i^1(t_i), & \text{if } t_i \in T_i^1 \\ b_i^2(t_i), & \text{if } t_i \in T_i^2 \end{cases}$$

for all types $t_i \in T_i$. Hence, $\mathcal{T}$ is a "block" type structure in which types in T_i^1 only refer to opponents' types in T_{-i}^1, and types in T_i^2 only refer to opponents' types in T_{-i}^2. But it is still a well-defined type structure, and hence we can run the *Type Partitioning Procedure* for the "block" type structure $\mathcal{T}$, yielding partitions $\mathcal{P}_i^\infty$ of the sets $T_i = T_i^1 \cup T_i^2$ for every agent i. If $t_i^1 \in T_i^1$ and $t_i^2 \in T_i^2$ turn out to be in the same equivalence class of $\mathcal{P}_i^\infty$, then t_i^1 and t_i^2 induce the same belief hierarchy. Otherwise not. In this way, the *Type Partitioning Procedure* can also be used to test whether two types from different type structures induce the same belief hierarchy or not.

Example 2. To see how this works, let us consider an example with two agents, $I = \{1, 2\}$, where the basic spaces of uncertainty are again given by $X_1 = \{c, d\}$ and $X_2 = \{e, f\}$ – as in Example 1 – together the the discrete σ-algebras on these sets. Consider the two type structures $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ and $\mathcal{T}^2 = (T_i^2, b_i^2)_{i \in I}$ on $\mathcal{X}$ as given in Table 2. Note that type structure $\mathcal{T}^1$ is almost identical to the type structure in Example 1, except for the fact that we have added an extra type t_1''' for agent 1.

We want to test whether the types $t_1 \in T_1^1$ and $r_1 \in T_1^2$, which belong to different type structures, induce the same belief hierrachy or not. As a first step we merge the two type structures $\mathcal{T}^1$ and $\mathcal{T}^2$ into one large block type structure, as described above. If we subsequently run the *Type Partitioning Procedure* for the large type structure, then the reader may verify that the partitions in every round are given by Table 3. Here, the procedure terminates at round 3. As t_1 and r_1 are not in the same equivalence class, we conclude that t_1 and r_1 do not induce the same belief hierarchy. In fact, the final partitions tell us that for agent 1, the types t_1, t_1', r_1' and r_1'' all induce the same belief hierarchy, that types t_1'' and r_1 induce the same belief hierarchy, and that for type $t_1''' \in T_1^1$ there is no type in T_1^2 that induces the same belief hierarchy. For agent 2, the types t_2, t_2'' and r_2' all induce the same belief hierarchy, and so do the types t_2', r_2 and r_2''. □

3.4 Testing for Properties of Type Structures

The *Type Partitioning Procedure* can be used to answer several different questions – local and global. First, as we already discussed, we can use it to test whether two types – possibly from different type structures – induce the same belief hierarchy or not. This is a local test.

Type structure $\mathcal{T}^1 = (T_1^1, T_2^1, b_1^1, b_2^1)$

$T_1^1 = \{t_1, t_1', t_1'', t_1'''\}, \qquad T_2^1 = \{t_2, t_2', t_2''\}$

$b_1^1(t_1) = \frac{1}{2}(c, t_2) + \frac{1}{2}(d, t_2')$
$b_1^1(t_1') = \frac{1}{6}(c, t_2) + \frac{1}{3}(c, t_2'') + \frac{1}{2}(d, t_2')$
$b_1^1(t_1'') = \frac{1}{2}(c, t_2') + \frac{1}{2}(d, t_2'')$
$b_1^1(t_1''') = \frac{1}{3}(c, t_2) + \frac{2}{3}(d, t_2'')$

$b_2^1(t_2) = \frac{1}{4}(e, t_1) + \frac{1}{2}(e, t_1') + \frac{1}{4}(f, t_1'')$
$b_2^1(t_2') = \frac{1}{8}(e, t_1) + \frac{1}{8}(e, t_1') + \frac{3}{4}(f, t_1'')$
$b_2^1(t_2'') = \frac{3}{8}(e, t_1) + \frac{3}{8}(e, t_1') + \frac{1}{4}(f, t_1'')$

Type structure $\mathcal{T}^2 = (T_1^2, T_2^2, b_1^2, b_2^2)$

$T_1^2 = \{r_1, r_1', r_1''\}, \qquad T_2^2 = \{r_2, r_2', r_2''\}$

$b_1^2(r_1) = \frac{1}{4}(c, r_2) + \frac{1}{4}(c, r_2'') + \frac{1}{2}(d, r_2')$
$b_1^2(r_1') = \frac{1}{2}(c, r_2') + \frac{1}{8}(d, r_2) + \frac{3}{8}(d, r_2'')$
$b_1^2(r_1'') = \frac{1}{2}(c, r_2') + \frac{3}{8}(d, r_2) + \frac{1}{8}(d, r_2'')$

$b_2^2(r_2) = \frac{1}{4}(e, r_1') + \frac{3}{4}(f, r_1)$
$b_2^2(r_2') = \frac{3}{4}(e, r_1') + \frac{1}{4}(f, r_1)$
$b_2^2(r_2'') = \frac{1}{8}(e, r_1') + \frac{1}{8}(e, r_1'') + \frac{3}{4}(f, r_1)$

Table 2: The type structures from Example 2

Round n	$\mathcal{P}_1^n$	$\mathcal{P}_2^n$
0	$\{\{t_1, t_1', t_1'', t_1''', r_1, r_1', r_1''\}\}$	$\{\{t_2, t_2', t_2'', r_2, r_2', r_2''\}\}$
1	$\{\{t_1, t_1', t_1'', r_1, r_1', r_1''\}, \{t_1'''\}\}$	$\{\{t_2, t_2'', r_2'\}, \{t_2', r_2, r_2''\}\}$
2	$\{\{t_1, t_1', r_1', r_1''\}, \{t_1'', r_1\}, \{t_1'''\}\}$	$\{\{t_2, t_2'', r_2'\}, \{t_2', r_2, r_2''\}\}$
3	$\{\{t_1, t_1', r_1', r_1''\}, \{t_1'', r_1\}, \{t_1'''\}\}$	$\{\{t_2, t_2'', r_2'\}, \{t_2', r_2, r_2''\}\}$

Table 3: The *Type Partitioning Procedure* in Example 2

But we can also use it to test global properties of type structures. For instance, we can use the procedure to test whether a given type structure contains *redundant* types or not – where "redundant" means that two different types induce the same belief hierarchy. For this redundancy test we can run the *Type Partitioning Procedure* and see whether the final partitions contain equivalence classes with at least two types. If this is the case then we conclude that the type structure contains redundancies. If, on the other hand, all equivalence classes contain only one type, then there are no redundancies in the type structure.

In case the type structure contains redundant types, the *Type Partitioning Procedure* will also tell us how to "remove" these redundancies without changing the induced collection of belief hierarchies. What we can do in this case is to replace every equivalence class in the final partition by a single type, and to change the belief of every type accordingly. Then we will obtain a smaller, non-redundant type structure that induces exactly the same collection of belief hierarchies.

As an illustration, consider the type structure from Table 1. We have seen in Example 1 that the *Type Partitioning Procedure* generates the final partitions

$$\mathcal{P}_1^\infty = \{\{t_1, t_1'\}, \{t_1''\}\} \text{ and } \mathcal{P}_2^\infty = \{\{t_2, t_2''\}, \{t_2'\}\}.$$

If we replace the equivalence class $\{t_1, t_1'\}$ by the single type r_1, and replace the equivalence class $\{t_2, t_2''\}$ by the single type r_2, then we obtain the smaller type structure $\hat{\mathcal{T}} = (\hat{T}_i, \hat{b}_i)_{i \in I}$ where

$$\hat{T}_1 = \{r_1, t_1''\},\ \hat{T}_2 = \{r_2, t_2'\}$$

and

$$\begin{aligned}
\hat{b}_1(r_1) &= \tfrac{1}{2}(c, r_2) + \tfrac{1}{2}(d, t_2'), \\
\hat{b}_2(t_1'') &= \tfrac{1}{2}(c, t_2') + \tfrac{1}{2}(d, r_2), \\
\hat{b}_2(r_2) &= \tfrac{3}{4}(e, r_1) + \tfrac{1}{4}(f, t_1''), \\
\hat{b}_2(t_2') &= \tfrac{1}{4}(e, r_1) + \tfrac{3}{4}(f, t_1'').
\end{aligned}$$

It can be verified that $\hat{\mathcal{T}}$ is indeed non-redundant, and induces the same collection of belief hierarchies as the original type structure $\mathcal{T}$ from Table 1.

Another global question we can answer is whether two different type structures generate the same *collection* of belief hierarchies, or whether the collection of belief hierarchies induced by the first type structure is contained in that of the second structure. This is the type of question which is addressed, for instance, in Friedenberg and Meier (2011). To answer the first question we can first merge the two type structures into one, and subsequently run the *Type Partitioning Procedure.* If the final partitions are such that every equivalence class always contains at least one type from both type structures, then the two structures generate the same collection of belief hiearchies. Otherwise not. Indeed, assume that every equivalence class in the final partitions

contains at least one type from each type structure. Then, for every type in the first structure there is a type in the second structure that generates the same belief hierarchy, and *vice versa.* That is, both type structures produce exactly the same collection of belief hierarchies. If this is not the case, that is, if there is an equivalence class that contains only types from one type structure but not from the other, then these type do not have any "counterpart" in the other type structure, and hence the two type structures differ in the collection of belief hierarchies they generate. To answer the second question – that is, whether the set of belief hierarchies of the first structure is contained in that of the second – we look at the final partitions, and see whether every equivalence class contains at least one type from the second structure.

We have collected the insights above in the following corollary.

Corollary 2 (Testing for Properties of Type Structures) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, and two finite type structures $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ and $\mathcal{T}^2 = (T_i^2, b_i^2)$ for $\mathcal{X}$. Let $(\mathcal{P}_i^\infty)_{i \in I}$ be the final partitions generated by the Type Partitioning Procedure if we first merge the two type structures into one. Then:*

(a) type structure $\mathcal{T}^1$ is redundant, if and only if, there is some agent i and some $P_i \in \mathcal{P}_i^\infty$ such that $|P_i \cap T_i^1| \geq 2$;

(b) the collection of belief hierarchies induced by $\mathcal{T}^1$ is a subset of the collection of belief hierarchies induced by $\mathcal{T}^2$, if and only if, $P_i \cap T_i^2 \neq \emptyset$ for all agents i and all $P_i \in \mathcal{P}_i^\infty$;

(c) type structures $\mathcal{T}^1$ and $\mathcal{T}^2$ induce the same collection of belief hierarchies, if and only if, $P_i \cap T_i^1 \neq \emptyset$ and $P_i \cap T_i^2 \neq \emptyset$ for all agents i and all $P_i \in \mathcal{P}_i^\infty$.

Here, we say that a type structure is redundant if it contains at least two different types that generate the same belief hierarchy.

4 Type and Hierarchy Morphisms

In this section we will relate the final partitions in the *Type Partitioning Procedure* to the notions of *type morphisms* and *hierarchy morphisms* in the literature, which are functions that map types from one type structure to types from another type structure while preserving the belief hierarchies. In the first part, we show that the final partitions in the *Type Partitioning Procedure* give rise to a generalized version of type morphisms – which we call *set-valued type morphisms.* We go on by demonstrating how set-valued type morphisms can be used to characterize types – possibly from different type structures – that share the same belief hierarchy. In the last part we discuss the relationship with hierarchy morphisms.

4.1 "Traditional" Type Morphisms

The notion of a *type morphism* has a long tradition in the literature on belief hierarchies and type structures. Informally, a type morphism is a function that maps the types from one type

structure to "placeholders" in the other type structure, such that the beliefs of the types about the opponents' choice-type conbinations are "preserved" through this mapping. The formal definition, which we present below, has – to the best of my knowledge – first been introduced by Böge and Eisele (1979), but also appears in Mertens and Zamir (1985), Heifetz and Samet (1998), Ely and Pęski (2006), Dekel, Fudenberg and Morris (2007), Liu (2009), Friedenberg and Meier (2011) and Pintér (2011), among others.

Definition 2 (Type Morphism) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, and two finite type structures $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ and $\mathcal{T}^2 = (T_i^2, b_i^2)$ for $\mathcal{X}$. A **type morphism** from $\mathcal{T}^1$ to $\mathcal{T}^2$ is a collection $(f_i)_{i \in I}$ of functions $f_i : T_i^1 \to T_i^2$ such that for all agents i, all types $t_i^1 \in T_i^1$, all $E_i \in \Sigma_i$, and all $t_{-i}^1 \in T_{-i}^1$,*

$$b_i^1(t_i^1)(E_i \times f_{-i}^{-1}(f_{-i}(t_{-i}^1))) = b_i^2(f_i(t_i^1))(E_i \times \{f_{-i}(t_{-i}^1))\}), \tag{3}$$

where $f_{-i}^{-1}(f_{-i}(t_{-i}^1)) := \{\hat{t}_{-i}^1 \in T_{-i}^1 \mid f_{-i}(\hat{t}_{-i}^1) = f_{-i}(t_{-i}^1)\}$.

In this definition, f_{-i} is the induced function from T_{-i}^1 to T_{-i}^2 which assigns to every $t_{-i}^1 = (t_j^1)_{j \neq i}$ in T_{-i}^1 the type combination $f_{-i}(t_{-i}^1) := (f_j(t_j^1))_{j \neq i}$ in T_{-i}^2.

Note that a type morphism does not make direct reference to belief hierarchies. An important property of type morphisms, however, is that they preserve the belief hierarchies of the types. That is, if $(f_i)_{i \in I}$ is a type morphism from the type structure $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ to the type structure $\mathcal{T}^2 = (T_i^2, b_i^2)$, then $f_i(t_i^1)$ always induces the same belief hierarchy as t_i^1 for all types $t_i^1 \in T_i^1$. See Proposition 5.1 in Heifetz and Samet (1998) for a formal proof. So, in particular, if $t_i^1 \in T_i^1$ and $t_i^2 \in T_i^2$ are two types from two – possibly different – type structures, and there is a type morphism between the two structures that maps t_i^1 to t_i^2, then both types share the same belief hierarchy.

The opposite direction is not true, however. To see this, consider the type structure $\mathcal{T}$ from Table 1. We have shown in Example 1 – by using the *Type Partioning Procedure* – that types t_1 and t_1' induce the same belief hierarchy. However, we will show that there is no type morphism from $\mathcal{T}$ to itself which maps t_1 to t_1', or t_1' to t_1.

To see this, suppose that $f = (f_1, f_2)$ is a type morphism from $\mathcal{T}$ to $\mathcal{T}$. As t_1 and t_1' induce the same belief hierarchy, which is different from the belief hierarchy induced by t_1'', we must have that $f_1(t_1), f_1(t_1') \in \{t_1, t_1'\}$ and $f_1(t_1'') = t_1''$. We consider the following three cases.

Case 1. Suppose that $f_1(t_1) = t_1'$ and $f_1(t_1') = t_1'$. Then,

$$b_2(t_2)(\{e\} \times f_1^{-1}(f_1(t_1))) = b_2(t_2)(\{e\} \times f_1^{-1}(t_1')) = b_2(t_2)(\{e\} \times \{t_1, t_1'\}) = \tfrac{3}{4}.$$

Hence, by (3) we must have that

$$b_2(f_2(t_2))(\{e\} \times f_1(t_1)) = b_2(f_2(t_2))(\{e\} \times \{t_1'\}) = \tfrac{3}{4},$$

which is impossible since there is no type $\tau_2 \in T_2$ with $b_2(\tau_2)(\{e\} \times \{t_1'\}) = \frac{3}{4}$. So, it cannot be that $f_1(t_1) = t_1'$ and $f_1(t_1') = t_1'$.

Case 2. Suppose that $f_1(t_1) = t_1'$ and $f_1(t_1') = t_1$. Then,

$$b_2(t_2)(\{e\} \times f_1^{-1}(f_1(t_1))) = b_2(t_2)(\{e\} \times f_1^{-1}(t_1')) = b_2(t_2)(\{e\} \times \{t_1\}) = \tfrac{1}{4}.$$

Hence, by (3) we must have that

$$b_2(f_2(t_2))(\{e\} \times f_1(t_1)) = b_2(f_2(t_2))(\{e\} \times \{t_1'\}) = \tfrac{1}{4},$$

which is impossible since there is no type $\tau_2 \in T_2$ with $b_2(\tau_2)(\{e\} \times \{t_1'\}) = \frac{1}{4}$. So, it cannot be that $f_1(t_1) = t_1'$ and $f_1(t_1') = t_1$.

Case 3. Suppose that $f_1(t_1) = t_1$ and $f_1(t_1') = t_1$. Then,

$$b_2(t_2)(\{e\} \times f_1^{-1}(f_1(t_1))) = b_2(t_2)(\{e\} \times f_1^{-1}(t_1)) = b_2(t_2)(\{e\} \times \{t_1, t_1'\}) = \tfrac{3}{4}.$$

Hence, by (3) we must have that

$$b_2(f_2(t_2))(\{e\} \times f_1(t_1)) = b_2(f_2(t_2))(\{e\} \times \{t_1\}) = \tfrac{3}{4},$$

which is impossible since there is no type $\tau_2 \in T_2$ with $b_2(\tau_2)(\{e\} \times \{t_1\}) = \frac{3}{4}$. Hence, it cannot be that $f_1(t_1) = t_1$ and $f_1(t_1') = t_1$.

But then, it must be the case that $f_1(t_1) = t_1$, $f_1(t_1') = t_1'$ and $f_1(t_1'') = t_1''$. In particular, there is no type morphism from $\mathcal{T}$ to $\mathcal{T}$ which maps t_1 to t_1', or t_1' to t_1. In fact, the reader may verify that the only type morphism from $\mathcal{T}$ to $\mathcal{T}$ is the identity mapping.

This example thus shows the following: If we want to test whether types t_i and t_i' – possibly from two different type structures – induce the same belief hierarchy, then the existence of a type morphism mapping t_i to t_i' is a *sufficient* condition, but not a *necessary* condition. Intuitively, the problem lies in the requirement that every type from the first type structure must be mapped to *exactly one* "placeholder" type from the second type structure. This requirement may be much too strong for testing whether t_i and t_i' share the same belief hierachy.

As we will show in the following subsection, the final partitions generated by the *Type Partitioning Procedure* give rise to a *generalization* of a type morphism, in which every type from the first type structure is not necessarily mapped to a *single* type from the other structure, but rather to a *set* of types from the other structure, which may be empty. We will refer to this generalized notion as a *set-valued type morphism.* We will show that the existence of a set-valued type morphism mapping t_i to (a set including) t_i' is both necessary and sufficient for checking whether t_i and t_i' induce the same belief hierarchy.

4.2 Set-valued Type Morphisms

Consider two finite type structures $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ and $\mathcal{T}^2 = (T_i^2, b_i^2)$ for the multi-agent uncertainty space $\mathcal{X}$. Suppose we apply the *Type Partitioning Procedure* to $\mathcal{T}^1$ and $\mathcal{T}^2$ by first merging the two type structures into one large type structure $\mathcal{T} = (T_i, b_i)_{i \in I}$ with $T_i = T_i^1 \cup T_i^2$ for all agents i. For every agent i, let $\mathcal{P}_i^\infty$ be the final partition of $T_i^1 \cup T_i^2$ generated by the procedure. Then, according to equation (2) in the procedure,

$$\mathcal{P}_i^\infty(t_i) = \{t_i' \in T_i \mid b_i(t_i')(E_i \times P_{-i}^\infty) = b_i(t_i)(E_i \times P_{-i}^\infty) \text{ for all } E_i \in \Sigma_i, \text{ and all } P_{-i}^\infty \in \mathcal{P}_{-i}^\infty\} \tag{4}$$

for all agents i and all $t_i \in T_i$.

The partition $\mathcal{P}_i^\infty$ induces, in a natural way, a *correspondence* $F_i : T_i^1 \twoheadrightarrow T_2^2$, assigning to every type $t_i^1 \in T_i^1$ the *set* of types

$$F_i(t_i^1) := \mathcal{P}_i^\infty(t_i^1) \cap T_i^2, \tag{5}$$

which is a subset of T_i^2. Note that $F_i(t_i^1)$ may be empty, since $\mathcal{P}_i^\infty(t_i^1)$ may contain no types from T_i^2. This is exactly the case when there is no type in T_i^2 inducing the same belief hierarchy as t_i^1. Also, $F_i(t_i^1)$ may contain more than one type, which is the case when there are several types in T_i^2 inducing the same belief hierarchy as t_i^1.

Let us now explore some properties of this correspondence F_i. Note first that, for every two different types $t_i^1, r_i^1 \in T_i^1$, either $F_i(t_i^1) = F_i(r_i^1)$, or $F_i(t_i^1) \cap F_i(r_i^1) = \emptyset$. We say that the correspondence F_i is *disjoint.*

For later purposes we define

$$F_i^{-1}(F_i(t_i^1)) := \{r_i^1 \in T_i^1 \mid F_i(r_i^1) = F_i(t_i^1)\}$$

for every $t_i^1 \in T_i^1$. Then, it may be verified that

$$F_i^{-1}(F_i(t_i^1)) = \mathcal{P}_i^\infty(t_i^1) \cap T_i^1 \tag{6}$$

for every $t_i^1 \in T_i^1$.

As expected, we define by F_{-i} the induced disjoint correspondence $F_{-i} : T_{-i}^1 \twoheadrightarrow T_{-i}^2$ which assigns to every $t_{-i}^1 = (t_j^1)_{j \neq i}$ in T_{-i}^1 the set

$$F_{-i}(t_{-i}^1) := \times_{j \neq i} F_j(t_j^1).$$

From (6) it then immediately follows that

$$F_{-i}^{-1}(F_{-i}(t_{-i}^1)) = \mathcal{P}_{-i}^\infty(t_{-i}^1) \cap T_{-i}^1 \tag{7}$$

for all $t_{-i}^1 \in T_{-i}^1$.

By combining (4), (5) and (7) we conclude that the collection of disjoint correspondences $(F_i)_{i \in I}$ has the following property: For every agent i, and every $t_i^1 \in T_i^1$,

$$F_i(t_i^1) = \{t_i^2 \in T_i^2 \mid b_i^1(t_i^1)(E_i \times F_{-i}^{-1}(F_{-i}(t_{-i}^1))) = b_i^2(t_i^2)(E_i \times F_{-i}(t_{-i}^1)) \text{ for all } E_i \in \Sigma_i, \text{ and all } t_{-i}^1 \in T_{-i}^1\}. \quad (8)$$

This is precisely the definition of a type morphism, except for the fact that the mappings F_i are now *correspondences* rather than *functions.* Indeed, if we assume that every correspondence F_i is in fact a *function* – that is, maps every type $t_i^1 \in T_i^1$ to a singleton set $\{t_i^2\}$ in T_i^2 – then condition (8) is identical to condition (3) in the definition of a type morphism. Condition (8) thus generalizes the notion of a type morphism to *correspondences* between sets of types. We will call this generalization a *set-valued type morphism.*

Definition 3 (Set-valued Type Morphism) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, and two finite type structures $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ and $\mathcal{T}^2 = (T_i^2, b_i^2)$ for $\mathcal{X}$. A* ***set-valued type morphism*** *from $\mathcal{T}^1$ to $\mathcal{T}^2$ is a collection $(F_i)_{i \in I}$ of disjoint correspondences $F_i : T_i^1 \twoheadrightarrow T_i^2$ such that for all agents i and all types $t_i^1 \in T_i^1$,*

$$F_i(t_i^1) = \{t_i^2 \in T_i^2 \mid b_i^1(t_i^1)(E_i \times F_{-i}^{-1}(F_{-i}(t_{-i}^1))) = b_i^2(t_i^2)(E_i \times F_{-i}(t_{-i}^1)) \text{ for all } E_i \in \Sigma_i, \text{ and all } t_{-i}^1 \in T_{-i}^1\}. \quad (9)$$

Note that a set-valued type morphism, like a "traditional" 'type morphism, does not make explicit reference to belief hierarchies. We have seen above that the final partitions $(\mathcal{P}_i^\infty)_{i \in I}$ of $T_i^1 \cup T_i^1$, generated by the *Type Partioning Procedure,* induce a set-valued type morphism $(F_i)_{i \in I}$ from $\mathcal{T}^1$ to $\mathcal{T}^2$. We will denote this set-valued type morphism by $F^{tpp} = (F_i^{tpp})_{i \in I}$.

As an illustration, consider the type structures $\mathcal{T}^1$ and $\mathcal{T}^2$ from Table 2. We have shown in Example 2 that the final partitions generated by the *Type Partitioning Procedure* are given by

$$\mathcal{P}_1^\infty = \{\{t_1, t_1', r_1', r_1''\}, \{t_1'', r_1\}, \{t_1'''\}\} \text{ and } \mathcal{P}_2^\infty = \{\{t_2, t_2'', r_2'\}, \{t_2', r_2, r_2''\}\},$$

(see Table 3). These partitions generate the set-valued type morphism $F = (F_1, F_2)$, with correspondences $F_1 : T_1^1 \twoheadrightarrow T_1^2$ and $F_2 : T_2^1 \twoheadrightarrow T_2^2$, where

$$\begin{aligned} F_1(t_1) &= F_1(t_1') = \{r_1', r_1''\}, \\ F_1(t_1'') &= \{r_1\}, \\ F_1(t_1''') &= \emptyset, \end{aligned}$$

$$\begin{aligned} F_2(t_2) &= F_2(t_2'') = \{r_2'\}, \\ F_2(t_2') &= \{r_2, r_2''\}. \end{aligned}$$

Note that $\mathcal{P}_1^\infty(t_1''') = \{t_1'''\}$ contains no type from T_1^2, because there is no type in $\mathcal{T}^2$ inducing the same belief hierarchy as t_1''', and therefore $F_1(t_1''') = \emptyset$.

4.3 Characterization Result

In this subsection we show how set-valued type morphisms can be used to characterize those groups of types that share the same belief hierarchy. As a first step towards this characterization, we start by showing that the set-valued type morphism F^{tpp} induced by the *Type Partioning Procedure* constitutes the "coarsest" possible set-valued type morphism. That is, we will prove that every set-valued type morphism must necessarily be a "refinement" of the set-valued type morphism F^{tpp}.

To formally define what we mean by a *refinement* of a set-valued type morphism $(F_i)_{i \in I}$, we will first naturally transform every disjoint correspondence F_i into a partition of $T_i^1 \cup T_i^2$, and will then relate the set-valued type morphisms by comparing these induced partitions. More precisely, consider a disjoint correspondence $F_i : T_i^1 \twoheadrightarrow T_i^2$. We define the induced partition $\mathcal{P}_i[F_i]$ of $T_i^1 \cup T_i^2$ by

$$(\mathcal{P}_i[F_i])(t_i) := \begin{cases} F_i^{-1}(F_i(t_i)) \cup F_i(t_i), & \text{if } t_i \in T_i^1 \text{ and } F_i(t_i) \neq \emptyset \\ \{t_i\}, & \text{if } t_i \in T_i^1 \text{ and } F_i(t_i) = \emptyset \\ \{t_i\}, & \text{if } t_i \in T_i^2 \text{ and } t_i \notin F_i(t_i^1) \text{ for any } t_i^1 \in T_i^1 \end{cases} \tag{10}$$

for every $t_i \in T_i^1 \cup T_i^2$. It can be verified that the collection of sets $\{(\mathcal{P}_i[F_i])(t_i) \mid t_i \in T_i^1 \cup T_i^2\}$ is indeed a partition of $T_i^1 \cup T_i^2$. Moreover, the reader can check that the partition $\mathcal{P}_i[F_i]$ induces exactly the disjoint correspondence F_i by means of the formula (5) above, and hence the partition $\mathcal{P}_i[F_i]$ really "belongs" to the correspondence F_i.

Now consider two disjoint correspondences F_i and F_i' from T_i^1 to T_i^2. We say that F_i is a *refinement* of F_i' if the induced partition $\mathcal{P}_i[F_i]$ of $T_i^1 \cup T_i^2$ is a refinement of $\mathcal{P}_i[F_i']$. Finally, if we compare two set-valued type morphisms $F = (F_i)_{i \in I}$ and $F' = (F_i')_{i \in I}$ from $\mathcal{T}^1$ to $\mathcal{T}^2$, we say that F is a refinement of F' if for every agent i, F_i is a refinement of F_i'.

We are now ready to formally state, and prove, the result which says that the set-valued type morphism induced by the *Type Partitioning Procedure* is the coarsest possible type morphism.

Theorem 3 (Coarsest Set-valued Type Morphism) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, and two finite type structures $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ and $\mathcal{T}^2 = (T_i^2, b_i^2)$ for $\mathcal{X}$. Let $F^{tpp} = (F_i^{tpp})_{i \in I}$ be the set-valued type morphism from $\mathcal{T}^1$ to $\mathcal{T}^2$ induced by the Type Partitioning Procedure. Then, every set-valued type morphism from $\mathcal{T}^1$ to $\mathcal{T}^2$ is a refinement of F^{tpp}.*

We will now use Theorem 3 to show that *set-valued* type morphisms, in contrast to "traditional" type morphisms, enable us to provide both sufficient *and necessary* conditions for two types inducing the same belief hierarchy.

Theorem 4 (Characterization by Set-valued Type Morphisms) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, and two finite type structures $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ and $\mathcal{T}^2 =$*

(T_i^2, b_i^2) *for* $\mathcal{X}$. *Take two types* $t_i^1 \in T_i^1$ *and* $t_i^2 \in T_i^2$. *Then,* t_i^1 *and* t_i^2 *induce the same belief hierarchy, if and only if, there is a set-valued type morphism* $(F_i)_{i \in I}$ *from* $\mathcal{T}^1$ *to* $\mathcal{T}^2$ *such that* $t_i^2 \in F_i(t_i^1)$.

In a sense, the above theorem shows what it takes to transform the traditional notion of a type morphism – which gives *sufficient,* but not necessary, conditions for two types sharing the same belief hierarchy – into a more "flexible" notion that yields both necessary and sufficient conditions.

4.4 Hierarchy Morphisms

We finally relate the output of the *Type Partitioning Procedure* to the notion of *hierarchy morphisms* as defined in Friedenberg and Meier (2011).

Definition 4 (Hierarchy Morphism) *Consider a multi-agent uncertainty space* $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, *and two finite type structures* $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ *and* $\mathcal{T}^2 = (T_i^2, b_i^2)$ *for* $\mathcal{X}$. *A* ***hierarchy morphism*** *from* $\mathcal{T}^1$ *to* $\mathcal{T}^2$ *is a collection* $(f_i)_{i \in I}$ *of functions* $f : T_i^1 \to T_i^2$ *such that* $h_i(f_i(t_i^1)) = h_i(t_i^1)$ *for all agents* i *and all* $t_i^1 \in T_i^1$.

That is, a hierarchy morphism is a function that maps types from the first structure into types from the second structure without changing the belief hierarchies. By means of Theorem 2 we can characterize the class of hierarchy morphisms from $\mathcal{T}^1$ to $\mathcal{T}^2$ as follows.

Corollary 3 (Characterization of Hierarchy Morphisms) *Consider a multi-agent uncertainty space* $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, *and two finite type structures* $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ *and* $\mathcal{T}^2 = (T_i^2, b_i^2)$ *for* $\mathcal{X}$. *Let* $(\mathcal{P}_i^\infty)_{i \in I}$ *be the final partitions generated by the Type Partitioning Procedure if we first merge the two type structures into one. Then, a collection* $(f_i)_{i \in I}$ *of functions* $f_i : T_i^1 \to T_i^2$ *is a hierarchy morphism, if and only if,* $f_i(t_i^1) \in \mathcal{P}_i^\infty(t_i^1)$ *for all agents* i *and all* $t_i^1 \in T_i^1$.

The proof follows directly from Theorem 2.

From Theorem 4 it follows, moreover, that every type morphism is a hierarchy morphism. Indeed, we have seen that every type morphism is a special case of a set-valued type morphism – where every type is mapped to a singleton set – and hence Theorem 4 guarantees that every type morphism preserves belief hierarchies. This result is also shown by Heifetz and Samet (1998).

The opposite direction, however, is not true: Not every hierarchy morphism is a type morphism. To see this, consider the type structure $\mathcal{T}$ from Table 1. We have seen in Example 1 that types t_1 and t_1' generate the same belief hierarchy. Therefore, the collection (f_1, f_2) of functions, where

$$f_1(t_1) = t_1', \;\; f_1(t_1') = t_1, \;\; f_1(t_1'') = t_1''$$

and f_2 is the identity mapping from T_2 to T_2, is a hierarchy morphism from $\mathcal{T}$ to $\mathcal{T}$. However, we have shown in Section 4.1 that there is no type morphism from $\mathcal{T}$ to $\mathcal{T}$ that maps t_1 to t_1' or

vice versa. In particular f, is not a type morphism. Friedenberg and Meier (2011) give another example where a hierarchy morphism is not a type morphism.

This raises the question: Under which circumstances is every hierarchy morphism also a type morphism? Friedenberg and Meier (2011) provide a full answer to this question by means of their Theorem 5.1. As a corollary of this result, they prove that every hierarchy morphism from $\mathcal{T}^1$ to $\mathcal{T}^2$ is a type morphism whenever $\mathcal{T}^2$ is non-redundant. See their Corollary 7.2, and bear in mind that we restrict to finite type structures. We can reproduce this result by relying solely on insights from this paper.

To see this, suppose that $\mathcal{T}^2$ is non-redundant. Let $(\mathcal{P}_i^\infty)_{i \in I}$ be the final partitions generated by the *Type Partitioning Procedure* if we first merge the two type structures $\mathcal{T}^1$ and $\mathcal{T}^2$ into one. Since $\mathcal{T}^2$ is non-redundant, we know from Corollary 2 that every equivalence class $P_i \in \mathcal{P}_i^\infty$ contains at most one type from T_i^2. Now, take an arbitrary hierarchy morphism $f = (f_i)_{i \in I}$ from $\mathcal{T}^1$ to $\mathcal{T}^2$. Then, by Corollary 3, $f_i(t_i^1) \in \mathcal{P}_i^\infty(t_i^1)$ for every $t_i^1 \in T_i^1$, and hence

$$\mathcal{P}_i^\infty(t_i^1) \cap T_i^2 = \{f_i^1(t_i^1)\} \text{ for all } t_i^1 \in T_i^1. \tag{11}$$

By (11), (4) and (6)it follows that $(f_i)_{i \in I}$ is a type morphism, as was to show. We have thus shown the following result.

Corollary 4 (Friedenberg and Meier (2011)) *Consider a multi-agent uncertainty space $\mathcal{X} = (X_i, \Sigma_i)_{i \in I}$, and two finite type structures $\mathcal{T}^1 = (T_i^1, b_i^1)_{i \in I}$ and $\mathcal{T}^2 = (T_i^2, b_i^2)$ for $\mathcal{X}$. Let $(\mathcal{P}_i^\infty)_{i \in I}$ be the final partitions generated by the Type Partitioning Procedure if we first merge the two type structures into one. Suppose that T^2 is non-redundant. Then, every hierarchy morphism from $\mathcal{T}^1$ to $\mathcal{T}^2$ is a type morphism.*

Friedenberg and Meier (2011) show in their Theorem 5.1 that – in a sense – the opposite direction is also true for finite type structures: If every hierarchy morphism from some (arbitrary) type structure to $\mathcal{T}^2$ is a type morphism, then $\mathcal{T}^2$ must be non-redundant. Consequently, if we consider finite type structures, then non-redundancy of $\mathcal{T}^2$ is both necessary and sufficient to guarantee that all hierarchy morphisms to $\mathcal{T}^2$ are also type morphisms. This follows from the fact that Friedenberg and Meier's condition of *strong measurability* – which plays a key role in their Theorem 5.1 – is equivalent to non-redundancy if the type structures are finite.

5 Possible Extensions

We will finally discuss some possible extensions of this work.

Infinite type spaces. An important restriction we impose in the present framework is that the type structures contain finitely many types only. But what if we allow the type structures to contain *infinitely* many types? Can we then still obtain characterization results *similar* to Theorems 2 and 4? One important thing that would change is that the *Type Partitioning Procedure*

is no longer guaranteed to terminate after finitely many steps, as there may be infinitely many successive strict refinements of partitions. Besides, the partitions produced at the various rounds of the procedure may contain infinitely many – even uncountably many – partition elements. On a more technical level, it also becomes important to investigate whether these partition elements constitute measurable sets or not, so that the procedure would still be well-defined. But even if these problems would all be sorted out, there still remains the question whether the *Type Partitioning Procedure* would eventually *characterize* those types that induce the same belief hierarchy. This seems an intriguing problem for future research.

From a conceptual viewpoint, how severe is the restriction that we restrict to finitely many types? This, of course, depends on the purpose one has in mind. It is well-known that many concepts in game theory, like *rationalizability* (Bernheim (1984), Pearce (1984)), *common belief in rationality* (Brandenburger and Dekel (1987), Tan and Werlang (1988)), *Nash equilibrium* (Nash (1950, 1951)), the *Dekel-Fudenberg procedure* (Dekel and Fudenberg (1990)), *proper rationalizability* (Schuhmacher (1999), Asheim (2001)) and *common belief in future rationality* (Perea (2014)) can be characterized by means of finite type structures. Perea (2012) shows that even concepts like *iterated assumption of rationality* (Brandenburger, Friendenberg and Keisler (2008)) and *common strong belief in rationality* (Battigalli and Siniscalchi (2002)), which are originally defined within complete – and hence infinite – type spaces, can also be characterized within finite type structures. Therefore, finite type structures would in principle suffice if the purpose is to investigate any of these concepts. But there may be other scenarios where infinite type structures turn out to be indispensable. For these scenarios it would then be crucial to have an analogue of Theorem 2 for the case where infinitely many types are allowed.

Finite belief hierarchies. In many situations of interest, it may simply be too demanding to require that players hold *infinitely* many levels of belief. It is therefore important to model type spaces in which certain types only hold beliefs up to a certain level n. See Kets (2010, 2013) and Heifetz and Kets (2013) for a thorough analysis of this phenomenon. An interesting question is whether the results in this paper can be extended to such settings where some types only induce a finite belief hierarchy.

Δ-*belief hierarchies.* Ely and Pęski (2006) introduce a new kind of belief hierarchy induced by types, which they call Δ-hierarchies. They proceed by showing that two types in an incomplete information setting induce the same set of *interim independent rationalizable* choices, if and only if, they induce the same Δ-hierarchy. As we already mentioned in a footnote in the introduction, this result is not true if we would consider "traditional" belief hierarchies rather than Δ-hierarchies. In fact, Δ-hierarchies contain more information than traditional belief hierarchies in the sense that two types sharing the same Δ-hierarchy will always share the same traditional belief hierarchy, but not *vice versa.* It would be interesting to see whether our approach can be used to examine Δ-hierarchies as well.

Alternative notions of belief. One could also try to extend the results in this paper to more general notions of belief, such as *lexicographic beliefs* (Blume, Brandenburger and Dekel (1991a,

1991b)) and *conditional beliefs* in dynamic games (Ben-Porath (1997), Battigalli and Siniscalchi (1999, 2002)). If one keeps the type structures finite, I expect that similar results – and similar proofs – should be possible for these settings as well.

Kripke-Aumann structures. Type structures are not the only way to encode belief hierarchies. One can also use models with states of the world, à la Kripke (1963) and Aumann (1976), to represent belief hierarchies. My feeling is that the results in this paper can be adapted to such models as well.

6 Proofs

Proof of Theorem 1. We first prove (a) by induction on n.

Induction start. The partition $\mathcal{P}_i^1$ will always be a refinement of $\mathcal{P}_i^0$ since $\mathcal{P}_i^0$ is the trivial partition, by definition.

Inductive step. Let $n \geq 2$, and suppose that $\mathcal{P}_i^{n-1}$ is a refinement of $\mathcal{P}_i^{n-2}$, for all agents i. Consider an agent i, an equivalence class $P_i^n \in \mathcal{P}_i^n$, and two types $t_i, t_i' \in P_i^n$. Then, by equation (2),

$$b_i(t_i)(E_i \times P_{-i}^{n-1}) = b_i(t_i')(E_i \times P_{-i}^{n-1}) \text{ for all } E_i \in \Sigma_i, \text{ and all } P_{-i}^{n-1} \in \mathcal{P}_{-i}^{n-1}. \quad (12)$$

As, by the induction assumption, $\mathcal{P}_j^{n-1}$ is a refinement of $\mathcal{P}_j^{n-2}$ for all $j \neq i$, it follows that $\mathcal{P}_{-i}^{n-1}$ is a refinement of $\mathcal{P}_{-i}^{n-2}$. But then, we conclude from (12) that

$$b_i(t_i)(E_i \times P_{-i}^{n-2}) = b_i(t_i')(E_i \times P_{-i}^{n-2}) \text{ for all } E_i \in \Sigma_i, \text{ and all } P_{-i}^{n-2} \in \mathcal{P}_{-i}^{n-2},$$

which means that t_i and t_i' belong to the same equivalence class in $\mathcal{P}_i^{n-1}$. So, we have shown that every two types that are in the same equivalence class of $\mathcal{P}_i^n$, are also in the same equivalence class of $\mathcal{P}_i^{n-1}$. This, however, implies that $\mathcal{P}_i^n$ is a refinement of $\mathcal{P}_i^{n-1}$, as was to show. By induction on n, property (a) follows.

With property (a) at hand, it is easy to prove property (b). By property (a) we know that for every round $n \geq 1$, and every agent i, the partition $\mathcal{P}_i^n$ is a refinement of $\mathcal{P}_i^{n-1}$. Moreover, for every "active" round n – where the procedure does not terminate yet – the partition $\mathcal{P}_i^n$ must be a *strict* refinement of $\mathcal{P}_i^{n-1}$ for at least one agent i. It may be verified that for every agent i, the number of successive strict refinements cannot be larger than the number of types in T_i. As such, the number of active rounds in the procedure cannot be larger than the number of types in $\cup_{i \in I} T_i$, which is N. This completes the proof. ■

Proof of Theorem 2. We first prove (a) by induction on n.

Induction start. Consider two types $t_i, t_i' \in T_i$. Suppose first that $h_i^1(t_i) = h_i^1(t_i')$. We show that $t_i' \in \mathcal{P}_i^1(t_i)$. For all $E_i \in \Sigma_i$ and $P_{-i}^0 \in \mathcal{P}_{-i}^0$,

$$\begin{aligned} b_i(t_i)(E_i \times P_{-i}^0) &= b_i(t_i)(E_i \times T_{-i}) = h_i^1(t_i)(E_i) \\ &= h_i^1(t_i')(E_i) = b_i(t_i')(E_i \times T_{-i}) = b_i(t_i')(E_i \times P_{-i}^0), \end{aligned}$$

which indeed implies that $t_i' \in \mathcal{P}_i^1(t_i)$. Here, the first and fifth equality follow from the fact that $\mathcal{P}_{-i}^0$ is the trivial partition on T_{-i}, the second and fourth equality follow from the definition of $h_i^1(t_i)$ and $h_i(t_i')$, respectively, whereas the third equality follows from the assumption that $h_i^1(t_i) = h_i^1(t_i')$.

Assume next that $t_i' \in \mathcal{P}_i^1(t_i)$. We show that $h_i^1(t_i) = h_i^1(t_i')$. For all $E_i \in \Sigma_i$,

$$\begin{aligned} h_i^1(t_i)(E_i) &= b_i(t_i)(E_i \times T_{-i}) = b_i(t_i')(E_i \times T_{-i}) \\ &= h_i^1(t_i')(E_i), \end{aligned}$$

which indeed implies that $h_i^1(t_i) = h_i^1(t_i')$. Here, the first and third equality follow from the definition of $h_i^1(t_i)$ and $h_i^1(t_i')$, respectively, whereas the second equality follows from the assumption that $t_i' \in \mathcal{P}_i^1(t_i)$ and equation (2).

By the two steps above we may conclude that $h_i^1(t_i) = h_i^1(t_i')$, if and only if, $t_i' \in \mathcal{P}_i^1(t_i)$, as was to show.

Inductive step. Let $n \geq 2$, and assume that (a) holds for $n-1$ and all agents i. Consider an agent i and two types $t_i, t_i' \in T_i$. Suppose first that $h_i^n(t_i) = h_i^n(t_i')$. We show that $t_i' \in \mathcal{P}_i^n(t_i)$.

Consider some $P_{-i}^{n-1} \in \mathcal{P}_{-i}^{n-1}$. Take some arbitrary $t_{-i} \in P_{-i}^{n-1}$ and let $h_{-i}^{n-1} = h_{-i}^{n-1}(t_{-i})$. By the induction assumption, P_{-i}^{n-1} contains all type combinations in T_{-i} that induce the same combination of $(n-1)$-th order beliefs as t_{-i}. Remember from Section 2.2 that $T_{-i}[h_{-i}^{n-1}]$ denotes the set of type combinations in T_{-i} that induce h_{-i}^{n-1}. Then, we may conclude that $P_{-i}^{n-1} = T_{-i}[h_{-i}^{n-1}]$. For every $E_i \in \Sigma_i$ we then have that

$$\begin{aligned} b_i(t_i')(E_i \times P_{-i}^{n-1}) &= b_i(t_i')(E_i \times T_{-i}[h_{-i}^{n-1}]) = h_i^n(t_i')(E_i \times \{h_{-i}^{n-1}\}) \\ &= h_i^n(t_i)(E_i \times \{h_{-i}^{n-1}\}) = b_i(t_i)(E_i \times T_{-i}[h_{-i}^{n-1}]) = b_i(t_i)(E_i \times P_{-i}^{n-1}), \end{aligned}$$

which by equation (2) indeed implies that $t_i' \in \mathcal{P}_i^n(t_i)$. Here, the first and fifth equality follows from the insight above that $P_{-i}^{n-1} = T_{-i}[h_{-i}^{n-1}]$, the second and the fourth equality follow from the definition of $h_i^n(t_i')$ and $h_i^n(t_i)$, whereas the third equality follows from the assumption that $h_i^n(t_i) = h_i^n(t_i')$.

Suppose next that $t_i' \in \mathcal{P}_i^n(t_i)$. We show that $h_i^n(t_i) = h_i^n(t_i')$.

Take some arbitrary combination $h_{-i}^{n-1} \in h_{-i}^{n-1}(T_{-i})$ of $(n-1)$-th order beliefs that is obtained by at least one type combination in T_{-i}. By the induction assumption, there must be some

$P_{-i}^{n-1} \in \mathcal{P}_{-i}^{n-1}$ such that $P_{-i}^{n-1} = T_{-i}[h_{-i}^{n-1}]$. Then, for every $E_i \in \Sigma_i$,

$$\begin{aligned} h_i^n(t_i)(E_i \times \{h_{-i}^{n-1}\}) &= b_i(t_i)(E_i \times T_{-i}[h_{-i}^{n-1}]) = b_i(t_i)(E_i \times P_{-i}^{n-1}) \\ &= b_i(t_i')(E_i \times P_{-i}^{n-1}) = b_i(t_i')(E_i \times T_{-i}[h_{-i}^{n-1}]) = h_i^n(t_i')(E_i \times \{h_{-i}^{n-1}\}), \end{aligned}$$

which indeed implies that $h_i^n(t_i) = h_i^n(t_i')$. Here, the third equality follows from the assumption that $t_i' \in \mathcal{P}_i^n(t_i)$ and equation (2), whereas the other equalities follow exactly as above.

By the two steps above we may thus conclude that $h_i^n(t_i) = h_i^n(t_i')$, if and only if, $t_i' \in \mathcal{P}_i^n(t_i)$. By induction on n, statement (a) follows.

The proof of (b) follows immediately from (a) and property (b) in Theorem 1, which implies that the procedure terminates after finitely many rounds. This completes the proof. ■

Proof of Theorem 3. Let $F = (F_i)_{i \in I}$ be an arbitrary set-valued type morphism from $\mathcal{T}^1$ to $\mathcal{T}^2$. For every $n \geq 0$, let $F^n = (F_i^n)_{i \in I}$ be the set-valued type morphism induced by the partitions $(\mathcal{P}_i^n)_{i \in I}$ generated in round n of the *Type Partinioning Procedure*. We show, by induction on n, that F_i is a refinement of F_i^n for all agents i and all $n \geq 0$.

Induction start. By construction, $\mathcal{P}_i^0$ is the trivial partition of $T_i^1 \cup T_i^2$, and hence F_i is a refinement of F_i^0, for all agents i.

Inductive step. Let $n \geq 1$, and assume that F_i is a refinement of F_i^{n-1} for all agents i. Consider an agent i. We show that F_i is a refinement of F_i^n. Hence, we must show that the partition $\mathcal{P}_i[F_i]$ is a refinement of the partition $\mathcal{P}_i[F_i^n]$, which requires us to prove that every equivalence class in $\mathcal{P}_i[F_i]$ is a subset of an equivalence class in $\mathcal{P}_i[F_i^n]$.

Take some equivalence class $P_i \in \mathcal{P}_i[F_i]$. Then, according to (10) there are two possibilities: either $P_i = F_i^{-1}(F_i(t_i^1)) \cup F_i(t_i^1)$ for some $t_i^1 \in T_i^1$ with $F_i(t_i^1) \neq \emptyset$, or $P_i = \{t_i\}$ for some $t_i \in T_i^1 \cup T_i^2$. Clearly, every equivalence class of the second kind is a subset of some equivalence class in $\mathcal{P}_i[F_i^n]$.

Now, consider an equivalence class $P_i \in \mathcal{P}_i[F_i]$ of the first kind, where $P_i = F_i^{-1}(F_i(t_i^1)) \cup F_i(t_i^1)$ for some $t_i^1 \in T_i^1$ with $F_i(t_i^1) \neq \emptyset$. Clearly, $t_i^1 \in F_i^{-1}(F_i(t_i^1))$, and hence $P_i = (\mathcal{P}_i[F_i])(t_i^1)$. In order to show that P_i is a subset of some equivalence class in $\mathcal{P}_i[F_i^n]$, it thus suffices to show that $(\mathcal{P}_i[F_i])(t_i^1) \subseteq (\mathcal{P}_i[F_i^n])(t_i^1)$.

Take some $\hat{t}_i^1 \in T_i^1$ with $\hat{t}_i^1 \in (\mathcal{P}_i[F_i])(t_i^1)$, and some $t_i^2 \in T_i^2$ with $t_i^2 \in (\mathcal{P}_i[F_i])(t_i^1)$. (The latter is possible since $\mathcal{P}_i[F_i] \cap T_i^2 = F_i(t_i^1)$, which is assumed to be non-empty). We show that $\hat{t}_i^1, t_i^2 \in (\mathcal{P}_i[F_i^n])(t_i^1)$.

Since $P_i = F_i^{-1}(F_i(t_i^1)) \cup F_i(t_i^1)$ we conclude that $\hat{t}_i^1 \in F_i^{-1}(F_i(t_i^1))$ – which means that $F_i(\hat{t}_i^1) = F_i(t_i^1)$ – and that $t_i^2 \in F_i(t_i^1)$. As $F_i(\hat{t}_i^1) = F_i(t_i^1)$, it follows that $t_i^2 \in F_i(\hat{t}_i^1)$ also. Since $F = (F_i)_{i \in I}$ is a set-valued type morphism from $\mathcal{T}^1$ to $\mathcal{T}^2$, it follows by (9) that

$$b_i^1(t_i^1)(E_i \times F_{-i}^{-1}(F_{-i}(t_{-i}^1))) = b_i^2(t_i^2)(E_i \times F_{-i}(t_{-i}^1)) = b_i^1(\hat{t}_i^1)(E_i \times F_{-i}^{-1}(F_{-i}(t_{-i}^1))) \qquad (13)$$

for all $E_i \in \Sigma_i$ and all $t^1_{-i} \in T^1_{-i}$. As, by our induction assumption, F_j is a of refinement F_j^{n-1} for all agents $j \neq i$, it follows that F_{-i} is a refinement of F_{-i}^{n-1}. But then, from (13) we can conclude that

$$b_i^1(t_i^1)(E_i \times (F_{-i}^{n-1})^{-1}(F_{-i}^{n-1}(t_{-i}^1))) = b_i^2(t_i^2)(E_i \times F_{-i}^{n-1}(t_{-i}^1)) = b_i^1(\hat{t}_i^1)(E_i \times (F_{-i}^{n-1})^{-1}(F_{-i}^{n-1}(t_{-i}^1)))$$

for all $E_i \in \Sigma_i$ and all $t^1_{-i} \in T^1_{-i}$. Since F_{-i}^{n-1} is the correspondence induced by the partition $\mathcal{P}_{-i}^{n-1}$, the equation above immediately implies that

$$b_i^1(t_i^1)(E_i \times P_{-i}^{n-1}) = b_i^2(t_i^2)(E_i \times P_{-i}^{n-1}) = b_i^1(\hat{t}_i^1)(E_i \times P_{-i}^{n-1})$$

for all $E_i \in \Sigma_i$ and all $P_{-i}^{n-1} \in \mathcal{P}_{-i}^{n-1}$. By equation (2) in the *Type Partitioning Procedure,* this means that $\hat{t}_i^1, t_i^2 \in \mathcal{P}_i^n(t_i^1)$. Since $\mathcal{P}_i^n(t_i^1) = (\mathcal{P}_i[F_i^n])(t_i^1)$, we conclude that $\hat{t}_i^1, t_i^2 \in (\mathcal{P}_i[F_i^n])(t_i^1)$, which was to show.

We have thus shown that $(\mathcal{P}_i[F_i])(t_i^1) \subseteq (\mathcal{P}_i[F_i^n])(t_i^1)$. Since this holds for every $t_i^1 \in T_i^1$ with $F_i(t_i^1) \neq \emptyset$, it follows that every equivalence class in $\mathcal{P}_i[F_i]$ is a subset of an equivalence class in $\mathcal{P}_i[F_i^n]$. Hence, $\mathcal{P}_i[F_i]$ is a refinement of $\mathcal{P}_i[F_i^n]$, which implies that F_i is a refinement of F_i^n.

By induction on n we may thus conclude that for all n, the set-valued type morphism $F = (F_i)_{i \in I}$ is a refinement of $F^n = (F_i^n)_{i \in I}$. Since $F^{tpp} = F^N$ for some N, it directly follows that F is a refinement of F^{tpp}, as was to show. This completes the proof. ■

Proof of Theorem 4. Suppose first that t_i^1 and t_i^2 induce the same belief hierarchy. Then, by Theorem 2, $t_i^2 \in \mathcal{P}_i^\infty(t_i^1)$, where $\mathcal{P}_i^\infty$ is the final partition of $T_i^1 \cup T_i^2$ generated by the *Type Partitioning Procedure.* Let $F^{tpp} = (F_i^{tpp})_{i \in I}$ be the set-valued type morphism from $\mathcal{T}^1$ to $\mathcal{T}^2$ induced by these partitions. Then, $t_i^2 \in F_i^{tpp}(t_i^1)$.

Suppose next that there is a set-valued type morphism $F = (F_i)_{i \in I}$ from $\mathcal{T}^1$ to $\mathcal{T}^2$ such that $t_i^2 \in F_i(t_i^1)$. By Theorem 3 we know that F is a refinement of F^{tpp}, and hence, in particular, $t_i^2 \in F_i^{tpp}(t_i^1)$. This, in turn, means that $t_i^2 \in \mathcal{P}_i^\infty(t_i^1)$. By Theorem 2 we may then conclude that t_i^1 and t_i^2 induce the same belief hierarchy. This completes the proof. ■

References

[1] Asheim, G.B. (2001), Proper rationalizability in lexicographic beliefs, *International Journal of Game Theory* **30,** 453–478.

[2] Aumann, R.J. (1976), Agreeing to disagree, *Annals of Statistics* **4**, 1236–1239.

[3] Battigalli, P. and M. Siniscalchi (1999), Hierarchies of conditional beliefs and interactive epistemology in dynamic games, *Journal of Economic Theory* **88**, 188–230.

[4] Battigalli, P. and M. Siniscalchi (2002), Strong belief and forward induction reasoning, *Journal of Economic Theory* **106,** 356–391.

[5] Ben-Porath, E. (1997), Rationality, Nash equilibrium and backwards induction in perfect-information games, *Review of Economic Studies* **64,** 23–46.

[6] Bernheim, B.D. (1984), Rationalizable strategic behavior, *Econometrica* **52**, 1007–1028.

[7] Blume, L.E., Brandenburger, A. and E. Dekel (1991a), Lexicographic probabilities and choice under uncertainty, *Econometrica* **59**, 61–79.

[8] Blume, L.E., Brandenburger, A. and E. Dekel (1991b), Lexicographic probabilities and equilibrium refinements, *Econometrica* **59,** 81–98.

[9] Böge, W. and T.H. Eisele (1979), On solutions of bayesian games, *International Journal of Game Theory* **8**, 193–215.

[10] Brandenburger, A. (2007), The power of paradox: Some recent developments in interactive epistemology, *International Journal of Game Theory* **35,** 465–492.

[11] Brandenburger, A. and E. Dekel (1987), Rationalizability and correlated equilibria, *Econometrica* **55**, 1391–1402.

[12] Brandenburger, A., Friedenberg, A. and J. Keisler (2008), Admissibility in games, *Econometrica* **76**, 307–352.

[13] Dekel, E. and D. Fudenberg (1990), Rational behavior with payoff uncertainty, *Journal of Economic Theory* **52**, 243–267.

[14] Dekel, E., D. Fudenberg and S. Morris (2007), Interim correlated rationalizability, *Theoretical Economics* **2,** 15–40.

[15] Dekel, E. and M. Siniscalchi (2013), Epistemic game theory, Chapter prepared for *Handbook of Game Theory.*

[16] Ely, J.C. and M. Pęski (2006), Hierrachies of belief and interim rationalizability, *Theoretical Economics* **1,** 19–65.

[17] Friedenberg, A. and M. Meier (2011), On the relationship between hierarchy and type morphisms, *Economic Theory* **46,** 377–399.

[18] Harsanyi, J.C. (1962), Bargaining in ignorance of the opponent's utility function, *Journal of Conflict Resolution* **6**, 29–38.

[19] Harsanyi, J.C. (1967–1968), Games with incomplete information played by "bayesian" players, I–III, *Management Science* **14**, 159–182, 320–334, 486–502.

[20] Heifetz, A. and W. Kets (2013), Robust multiplicity with a grain of naiveté, Working paper.

[21] Heifetz, A. and D. Samet (1998), Topology-free typology of beliefs, *Journal of Economic Theory* **82,** 324–341.

[22] Kets, W. (2010), Bounded reasoning and higher-order uncertainty, Working paper, Northwestern University.

[23] Kets, W. (2013), Finite depth of reasoning and equilibrium play in games with incomplete information, Working paper, Northwestern University.

[24] Kripke, S. (1963), A semantical analysis of modal logic I: Normal modal propositional calculi, *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* **9**, 67–96.

[25] Liu, Q. (2009), On redundant types and Bayesian formulation of incomplete information, *Journal of Economic Theory* **144,** 2115–2145.

[26] Mertens, J.-F. and S. Zamir (1985), Formulation of bayesian analysis for games with incomplete information, *International Journal of Game Theory* **14**, 1–29.

[27] Nash, J.F. (1950), Equilibrium points in N-person games, *Proceedings of the National Academy of Sciences of the United States of America* **36,** 48–49.

[28] Nash, J.F. (1951), Non-cooperative games, *Annals of Mathematics* **54**, 286–295.

[29] Pearce, D. (1984), Rationalizable strategic behavior and the problem of perfection, *Econometrica* **52**, 1029–1050.

[30] Perea, A. (2012), *Epistemic Game Theory: Reasoning and Choice,* Cambridge University Press.

[31] Perea, A. (2014), Belief in the opponents' future rationality, *Games and Economic Behavior* **83,** 231–254.

[32] Pintér, M. (2011), Invariance under type morphisms: the Bayesian Nash equilibrium, Working paper.

[33] Schuhmacher, F. (1999), Proper rationalizability and backward induction, *International Journal of Game Theory* **28**, 599–615.

[34] Tan, T. and S.R.C. Werlang (1988), The bayesian foundations of solution concepts of games, *Journal of Economic Theory* **45**, 370–391.

[35] Weinstein, J. and M. Yildiz (2007), Impact of higher-order uncertainty, *Games and economic Behavior* **60,** 200–212.