Coordinate Systems: Level Ascending Ontological Options

Chris Partridge
BORO Solutions Ltd
University of Westminster
0000-0003-2631-1627

Andrew Mitchell
BORO Solutions Ltd
0000-0001-9131-722X

Michael Loneragan
QinetiQ Group PLC
Portsmouth, Hampshire, UK
mjloneragan@qinetiq.com

Hayden Atkinson
CooperVision Ltd
0000-0002-5153-9116

Sergio de Cesare
University of Westminster
0000-0002-2559-0567

Mesbah Khan
OntoLedgy Ltd
University of Westminster
0000-0002-1327-6263

"Philosophy [nature] is written in that great book whichever is before our eyes -- I mean the universe -- but we cannot understand it if we do not first learn the language and grasp the symbols in which it is written. The book is written in mathematical language, and the symbols are triangles, circles and other geometrical figures, without whose help it is impossible to comprehend a single word of it; without which one wanders in vain through a dark labyrinth."

— Galileo, Il Saggiatore, 1623

***Abstract*—A major challenge faced in the deployment of collaborating unmanned vehicles is enabling the semantic interoperability of sensor data. One aspect of this, where there is significant opportunity for improvement, is characterizing the coordinate systems for sensed position data. We are involved in a proof of concept project that addresses this challenge through a foundational conceptual model using a constructional approach based upon the BORO Foundational Ontology. The model reveals the characteristics as sets of options for configuring the coordinate systems. This paper examines how these options involve, ontologically, ascending levels. It identifies two types of levels, the well-known type levels and the less well-known tuple/relation levels.**

Keywords—BORO Foundational Ontology; Constructional Ontology; Geometric Coordinate System Ontology; Power-Type-Builder; Power-Tuple-Builder; Multi-Level Options; Multi-Platform-Domain Sensor System.

I. Introduction

It is well-recognised that a major challenge currently facing the deployment of collaborating unmanned vehicles is semantic interoperability [1, 2], and that as this technology develops, the requirements for interoperability are likely to become both more stringent and complex. A common (preferably open) data architecture is seen as key to resolving this [1, 2]. Many of the current vehicles use systems and data structures that are proprietary and have a single platform – single domain heritage. These typically made no use of conceptual models in their development, and so have a lightweight (sometimes, non-existent) conceptual framework. It is a situation with substantial opportunities for improvement [3].

In most vehicles, sensors are the major producers of data, with a significant proportion of this being sensed position data. Sensed positions (typically structured as a triple of coordinates) are relative to a coordinate system. Where there are multiple platforms-domains, their sensors will use different local coordinate systems. To be able to integrate this sensor data into a single common picture, the integrating system needs to know the various sensed positions' coordinate systems. For example, the integrating system might receive two sensed position triples from different platform's sensors with the same coordinate numerals (such as <10, 20, 30>). These would typically use different local coordinate systems and if the integrating system does not know which coordinate system each is relative to, it cannot interpret and integrate them. The '20' coordinate in the first triple might be relative to a Cartesian coordinate system and so refer to a distance and the coordinate in the second triple be relative to a Spherical coordinate system and so refer to an azimuthal angle – or maybe vice versa. In general, if the integrating system does not know enough about the 'owning' coordinate systems, it cannot interpret the position triples and so integrate them.

More generally, what is required is an understanding of which characteristics of the coordinate system need to be known so that the position triple can be interpreted. Unfortunately, little work has been done on determining what a full characterisation would look like. In practice, coordinate systems are often not explicitly defined at all: it is assumed that users of the sensors know enough about what their coordinate system is. Where coordinate systems are defined, the characterisation is partial and pragmatically ad hoc.

We are working on a Proof of Concept (PoC) project that is assessing a radical approach to developing a suitable conceptual architecture for articulating the full requirements. The aim is to uncover the underlying conceptual foundations revealing a clear fundamental picture and, in so doing, to strip away any pre-conceptions remaining from the single platform-domain heritage. To uncover its conceptual foundation, we took a close technical look at the geometric foundations of the world described by the sensor position data. The project's prime analytic tool is a constructional ontology based upon the BORO Foundational Ontology [4, 5].

Early work has focused on three simple coordinate systems for sensed positions and is clearly exposing an underlying compositional geometric structure; one where systems are built from a common set of ontological components whose construction processes follow broadly similar stages. In this paper, we focus on one aspect that is interesting from a multi-level modelling perspective. We investigate how to characterise the variety of coordinate systems as a series of sets of options. We focus on how these sets of options are embodied by generating objects at a higher level. As well as the well-known 'type' multi-levels (associated with Powertypes [6]), there is a second, less appreciated, kind of level-ascending based upon 'tuples/relations'. We characterise these here as combination (types) and permutation (tuples/relations) options. This seems to give us a full picture of the fundamental characteristics needed to interpret the position.

A. *Structure of this Paper*

The body of the paper is structured into four parts. The first part (Section II) aims to give a general context by briefly outlining the overall project that framed this work, then describing the specific coordinate system characteristics challenge and how we aim to address it. It then goes on to describe the initial PoC project that focuses on this specific challenge. The next three parts of the paper contain the technical analysis. The second part (Section III) provides a methodological background by describing the ontological framework that is being deployed to build the conceptual model – and includes simple examples to introduce the two kinds of options focused on in this paper. The third part (Section IV) provides an overview of the ontology as an introduction to the examples of the options as levels presented in the fourth part (Section V). A final summary concludes the paper.

II. The Project

The current PoC project showcases an approach to building a foundational conceptual model that should be capable of resolving the semantic integration problems that multi-platform/domain sensor systems are currently facing. In the following sections, we look at the context in more detail and then give an overview of the requirements of the project. We then note our insights and show how this motivates our approach.

A. *Context (In More Detail)*

Unmanned vehicle collaboration across multiple domains/environments (air, surface, land, underwater and space) is recognised as a difficult engineering problem - Fig. 1 shows examples of both single and multiple platform/domain manned and unmanned collaborating vehicles.

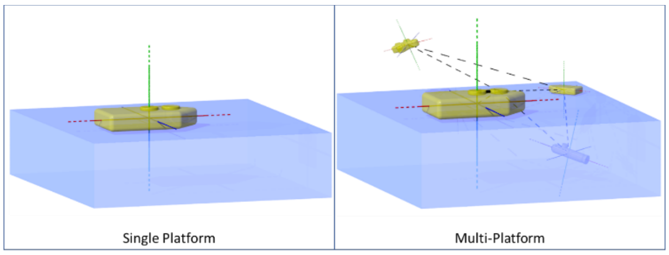

Fig. 1. Single and Multi-Platform with Deictic Axes

One challenge is the semantic integration of the sensing data into a single common picture – ground truth. A common data architecture with agreed data structures and APIs would simplify the challenge at the syntactic level. But this needs to be supported by a common semantic model to ensure shared semantics. The scope of such a model extends beyond the APIs, as their semantic integrity depends upon the systems behind them respecting (and so understanding) it.

The unmanned vehicle sensors process the raw data and pass this on to other, typically centralised, systems for further processing. The level of local onboard processing varies depending upon various factors; for example, low bandwidth restrictions might lead to a preference for onboard over centralised remote processing. The sensors work on a local basis of own position and measure other positions relative to themselves – they may further process these measurements before reporting or directly report a sensed position relative to themselves. Directly reporting the local positions may be preferred as this allows a centralised, consistent calculation of errors.

At the core of these reports is a sensed position recorded using coordinates. The data format of these coordinates is apparently simple and easy to specify – a triple of numbers, with a time-stamp. However, it has emerged that it is more of a challenge to find a common data (and semantic) format to characterise all the coordinate systems to which these coordinates can (or could) belong. In large part, this is because the common format will need to be able to accommodate significantly more variety and complexity than the current single platform-domain systems – and include enough detail to make coordinate conversions between the systems, or to a common system. The ways in which the systems vary include:

- *Coordinate system.* Unmanned vehicles should be easy to add to (and remove from) the collaborative sensor systems – whatever coordinate system they use. These vehicles are likely to use new types of coordinate systems which will need to be supported. So, some general structure for coordinate systems needs to be developed.

- *Position and orientation.* The unmanned platforms are moving (with both distance and angular velocity) relative to the main platform in all three dimensions – so the position and orientation of their coordinate systems will be both different and varying. So, some general structure for position and orientation needs to be developed.

- *Angle and unit.* There will typically be limited governance over suppliers of the unmanned vehicles, who are likely to use their own configuration for the coordinate systems. For example, they may use different distance units; one using kilometres, the other miles. So, some general structure for distance and angle units as deployed in coordinate systems needs to be developed.

- *Domain-specific simplifications.* Platforms in the sea domain have, in the past, often had more basic requirements than other domains. For example, they have typically used small-angle approximation, and some even only considering yaw angular movements, ignoring roll and pitch – as these are not so relevant for single platforms in the sea domain. ([7, 8] describe another simplification for position calculation.) More generally, this raises the requirement, in multi-domain systems, for these domain-specific simplifications to be harmonised to avoid error-generating inconsistencies.

- *Direction.* Different platforms and sensors will use different directions within the orientations. For example, the Cartesian z-axis often points downwards for underwater and aerial platforms and an up direction for surface platforms (in the maritime domain, this can vary from ship to ship). So, some general structure for directions as deployed in coordinate systems needs to be developed.

B. *The Project's Aims*

The PoC aims to build a conceptual model that will support the semantic unification requirements of multiple platform-domain systems. More generally, it aims to showcase a general methodology for designing the data architecture of this domain; one that involves a principled, repeatable, auditable, extendable process. Such a process should be able to identify the range of possible coordinate systems characteristics (possibly exposing their foundations) and design a parsimonious and elegant conceptual model for representing them.

This should provide a degree of comfort that the data architecture built from the conceptual model not only accurately covers current requirements but is also relatively future-proofed:

- that the process will identify a reasonably complete range of possible configurations and
- that it will be easily extendable to new coordinate systems.

It should also provide a benchmark for identifying gaps in the existing data architectures.

C. *Our Insights*

The following three insights motivated the approach for developing the conceptual framework outlined in this paper:

1. Each characteristic of the coordinate systems can be thought of as an exhaustive set of independent options. For example, the coordinate system's surface configuration type may be Cartesian, Cylindrical, or Spherical—one of these options needs to be selected. Generally, the sets of independent options seem to come in two varieties (kinds), combinations and permutations. These correspond, respectively, to ways the system can be and to ways of organising the system.
2. Currently, there is no obvious parsimonious and elegant framework for organising these characteristics waiting to be plucked off the shelf. Standards, such as [9] and [10], do not (upon inspection) provide the right kind of help. Neither does theoretical work such as [11]. Though, of course, all of these provide useful input. In some ways, this is a surprising situation, as Euclidean coordinate geometry has been researched extensively for millennia. In other ways, it is not so surprising, as the motivation for this research has not been to unearth the characteristics that should drive a conceptual model to support a data architecture.
3. The coordinate system characteristics that drive the conceptual model are grounded in the system's geometry and that an understanding of these characteristics will emerge from a clear picture of its foundational geometrical features. (As a side note, there is a revived interest in geometry as a mathematical foundation for space and time – see, for example [12] – as well as one in the foundations of Euclid's original geometric work – see, for example, [13])

D. *Our Approach*

We decided to start with an ontological conceptual model which would give us a technology agnostic picture. Given the importance of exposing the geometric foundations, we recognised the need to be geometry friendly. We choose a foundational ontology that is extensional and four-dimensional, the BORO Foundational Ontology [5], and deploy it using a constructional approach [4]. We expected this to not only expose the foundations of the range of possible coordinate systems characteristics but also provide a workspace for exploring the relative parsimony and elegance of different conceptual structures. We also adopted as a goal to understand what the ontology of the coordinate system options is, to enable us to use this to design the data architecture.

As a first stage, we started a PoC project for the limited set of the three simplest local coordinate systems; Cartesian (sometimes called Rectangular – though from our perspective Planar would be more accurate), Spherical and Cylindrical. We also assumed that we could simplify the geometry to Euclidean affine space-time. We build upon earlier work we have done with coordinate systems [14, 15]. This project is under way, and this paper is based upon the early results.

III. Ontological Framework

We have found it extremely useful to adopt a constructional approach, but it is a highly technical – so here and in the next section we only attempt a simple overview (sufficient for our purposes), pointing the interested reader to detailed expositions elsewhere. In this section, we introduce the general background ontological framework: in the next section we give an overview of the coordinate system ontology. We start by briefly introducing the BORO constructional ontology. We then give simple examples which clearly illustrate how options (or possibilities) arise ontologically from two key level-generating operations. And so, the more complex (real) examples in the next section should be easier to understand.

A. *BORO Constructional Ontology*

The BORO Foundational Ontology [5, 16] is an extensional four-dimensional ontology. Its metaphysical architecture (choices) are described in [17]. The constructional approach to building ontologies is highly technical and we recommend that readers interested in the details consult [4], which uses the BORO Ontology as an example. It is also well worth consulting Fine's papers [18, 19] upon which this approach is based.

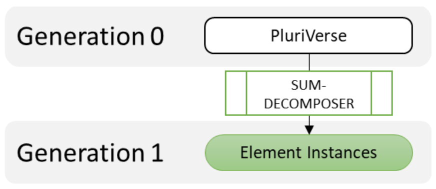

Fig. 2. BORO ONTOGENESIS – Generations 0 and 1

1) BORO ONTOGENESIS – Generations 0 and 1. [4] introduces the notion of ONTOGENESIS, the construction process for the ontology and the generational operations that compose it. This process is described in the example SIMPLE in [4]. The example is sufficiently expressive for our purposes here. In overview the process is as follows (again, see [4] for details). Constructional BORO requires some initial work to generate a collection for the POWER operations to work on. It takes as the initial START – generation 0 – the PluriVerse (the fusion of all possible worlds) based upon its adoption of priority monism [20]. The PluriVerse is then deconstructed using SUM-DECOMPOSER into all possible parts (the instances of BORO's Elements) – generation 1 – as shown diagrammatically in Fig. 2.

2) ONTOGENESIS$_C$. In the next sections, we describe this project's ontogenesis – called ONTOGENESIS$_C$ – looking at how the two POWER operations (POWER-TYPE-BUILDER and POWER-TUPLE-BUILDER) work over the generation 1 collection.

3) POWER-TYPE-BUILDER. The examples in [4] focused on the TYPE-BUILDER constructor (in that paper, following [19], this was called SET-BUILDER) and its POWER operation – which we call, to remain consistent with the BORO terminology, POWER-TYPE-BUILDER. The POWER operation for a constructor applies the constructor to all possible candidates from its input collection (of any size, finite or infinite) – exhausting the 'power' of the constructor. In the case of POWER-TYPE-BUILDER, the POWER operation for the TYPE-BUILDER constructor, it generates all the possible types by applying the TYPE-BUILDER constructor to all possible sub-collections of the input collection. These generated types then form part of the next level or generation.

The generation 1 collection (Fig. 2) provides the initial basis for POWER-TYPE-BUILDER generation. Applying POWER-TYPE-BUILDER to this generates all the instances of Elements Powertype – generation 2. Applying POWER-TYPE-BUILDER to this generation generates all the instances of Elements Powertype Powertype – generation 3. This is shown diagrammatically in Fig. 3 up to generation 4. Cumulatively, each application of the POWER-TYPE BUILDER adds a link to the chains of instantiation.

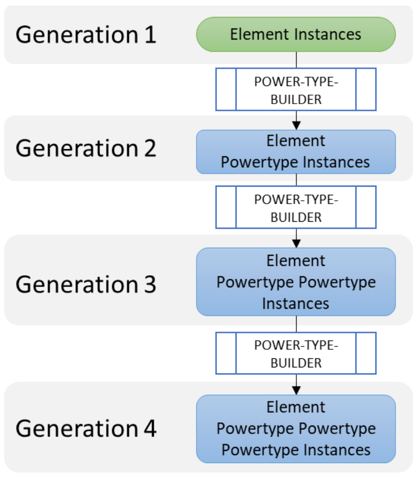

Fig. 3. ONTOGENESIS$_C$ - Pure POWER-TYPE-BUILDER Branch - Generations 2 to 4

Note: The 'Powertype' in BORO's Elements Powertype is set-theoretic, the set of all subsets of Elements. This is the result of applying TYPE-BUILDER to the complete output of the POWER-TYPE-BUILDER used to construct generation 2, which is done as part of the next generation.

4) POWER-TUPLE-BUILDER. [4] identified the atomic TUPLE-BUILDER constructor (in that paper it is called SEQUENCE-BUILDER following [19]) – again, see referenced papers for technical details. Its POWER operation was outside the scope of the paper's examples but is needed for this project, so we describe it here.

POWER-TUPLE-BUILDER applies the TUPLE-BUILDER constructor to all possible sequences (permutations) from sub-collections of the input collection. These generated tuples then form part of the next level of generation. We add the constructor TUPLE-BUILDER to our ONTOGENESIS$_C$ – with the additional operation, POWER-TUPLE-BUILDER, to build generations. This operation can be regarded as a generalisation of the mathematical notion of a Cartesian Product, which is restricted to a fixed number of tuple places.

In constructional BORO, POWER-TUPLE-BUILDER operation starts at the same stage as POWER-TYPE-BUILDER. Its first application is to the collection of all possible individuals that emerges at generation 1. Applying POWER-TUPLE-BUILDER to this generates all the instances of BORO's element tuples – generation 2. Note: 'element tuples' is the result of applying TYPE-BUILDER to the complete output of the POWER-TUPLE-BUILDER branch of generation 2, which is done as part of the next generation. More generally, to keep things as simple as required, POWER-TUPLE-BUILDER is only applied to the collections generated by pure (unmixed) POWER-TYPE-BUILDER generations. This is shown up to generation 4 in Fig. 4.

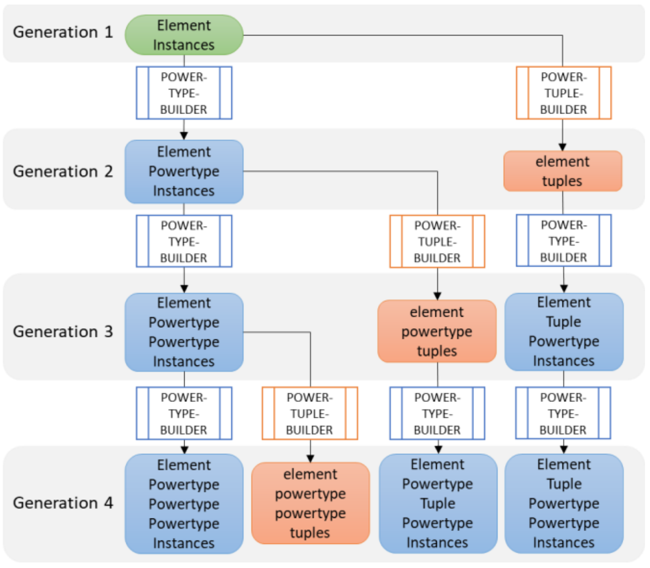

Fig. 4. ONTOGENESIS$_C$ with both POWER-TYPE-BUILDER and POWER-TUPLE-BUILDER

We understand that it is unusual to consider tuple levels: mainstream multi-level modelling does not consider tuple as a basis for level hierarchies. However, if one considers their (tuples) fundamental structure, they seem to have as much claim as types to be regarded as level generating. As Fine noted: "there is an intuitive distinction between wholes which are like sets in being hierarchically organised and those which are like sums in being 'flat', or without an internal division into levels. The distinction, under the operational approach, can be seen to turn on whether repeated applications of the operation are capable of yielding something new." [19, p.

566]. Given both types and tuples have this level-generating property, it seems natural to treat both as generating levels.

B. Options (or Ways of Arranging) as Levels

In the project, we have found two broad kinds of option that arise ontologically from these two level-generating operations; combination and permutation. We give simple, illustrative examples here to introduce the notion, to make it easier to understand the project-based examples described later. Though this association of levels with options may be uncommon in conceptual modelling, something similar is found in combinatorial mathematics, where combinations and permutations are well-known types of arrangements: hence these examples will look familiar to those found there.

1) Combination Options Example. Consider a game for two players that is played with just the four aces (the 'cards') from the standard deck of 52 playing cards. Assume that at the start of the game, someone deals these into two equal piles of two cards – two 'hands'. Let's say that hand one contains the aces of hearts and diamonds and hand two the other two aces, clubs and spades. We have talked about hands (piles), but what type of object could these be. An obvious candidate is the type (set) of the two cards – {hearts, diamonds}. If we want to talk about the way the cards have been dealt, the two hands (piles) that resulted – a 'deal' – then the obvious candidate is the set of the two sets – {{hearts, diamonds}, {clubs, spades}}. If we want to talk about all the possible ways the cards could have been dealt, all possible deals, then this would be the set of all three possible deals – shown in Fig. 5 – the deal combinations.

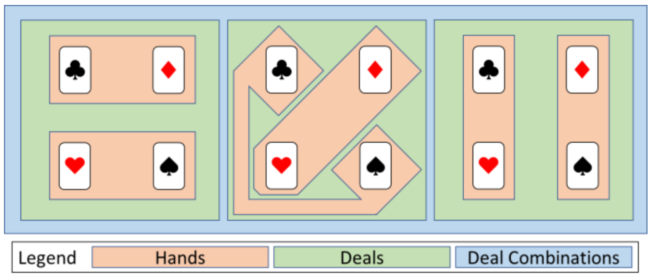

Fig. 5. Set of All Possible Deals – the Three Deal Combinations

From a constructional point of view (and simplifying a little), we start with the four aces and then apply TYPE-BUILDER to all combinations of two cards to give us the possible hands. We then apply TYPE-BUILDER to all combinations of two hands that are disjoint, to give us all (three) possible deals. We then apply TYPE-BUILDER to all the deals – which gives us the single object 'deal combinations'. When hands are dealt, then the deal will be one of these. The construction is shown diagrammatically in Fig. 6. Deal Combination is called a 'combination' (option) because it contains the different possible ways of combining hands.

The construction process makes clear how each object emerges from ascending a level in the ONTOGENESIS process. Individual *cards* emerge in generation 1, individual *hands* in generation 2, *deals* in generation 3 and the *deal combinations* in generation 4. This is shown diagrammatically in Fig. 6.

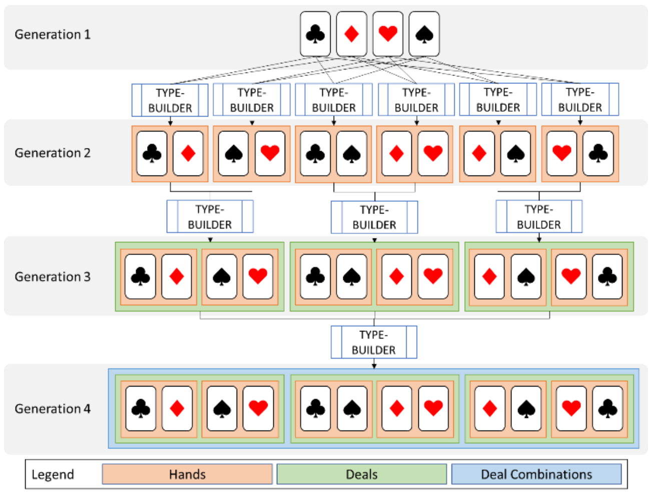

Fig. 6. Deal Combination Construction

2) Permutation Options Example. So far, we have not considered which player gets which hand. Reconsider the deal where hand one contains the aces of hearts and diamonds and hand two the other two aces, clubs and spades. What ways could these hands be distributed (permuted) across the two players – the player deals? Player A could have hand one and player B hand two – or vice versa. Clearly, there are two player deal permutation options, both with the same content but ordered in different ways – see Fig. 7. These options are called permutation options because they are ways of arranging the hands among the players. And, as the members of a set are not ordered, sets of hands cannot capture this permutation structure, but tuples *construction* can; the tuple <hand one, hand two> is different from the tuple <hand two, hand one>.

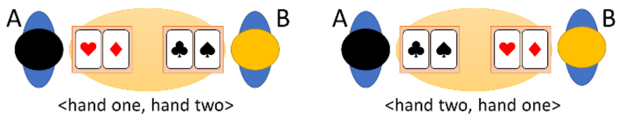

Fig. 7. Two Player Deal Permutation Options

We can construct these two players' deal permutations as tuples by taking the two instances of the deal type and applying TUPLE-BUILDER to the permutations of the two instances. More generally, we can create all the possible player deal permutations by taking every deal instance of deal combinations and then applying TUPLE-BUILDER to the two possible permutations of their two instances. If we apply TYPE-BUILDER to these tuple permutations, we get the object *player deal permutations*. See Fig. 8.

The constructional approach shows clearly how the individual permutations are generated by TUPLE-BUILDER – and so how permutations involve ascending tuple levels.

As before, the construction process makes clear how the objects emerge from ascending a tuple level in the ONTOGENESIS process; in this case, tuple permutations. Individual *cards* emerge in generation 1, individual *hands* in generation 2, *player deals* in generation 3 (via POWER-TUPLE-BUILDER) and the *player deal permutations* in generation 4. This is shown diagrammatically in Fig. 8.

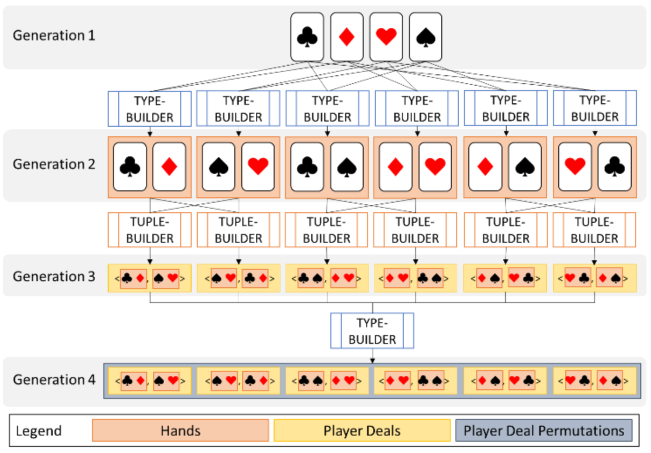

Fig. 8. Player Deal Permutations

IV. The Coordinate System Constructional Ontology

In this section, we provide a brief simplified overview of the overall coordinate system constructional ontology as a context for the multi-level option examples in the next section. We hope to provide a fuller description of this ontology in future papers.

A. *Analysis Through Geometric Construction*

The analysis is a kind of logical construction in the spirit of Bertrand Russell ("Wherever possible, substitute constructions out of known entities for inferences to unknown entities." [21, p. 363]) and Rudolf Carnap [22]. It involves a search for both the grounding geometric objects and how these are constructed. It usually takes substantial analysis and experimentation to identify suitable grounding components and associated elegant and parsimonious construction processes.

It became clear from early in the analysis that there were two sub-ontologies in play for this topic. The 'pure' coordinate system ontology and a deixis ontology that explains how the platform is, in practice, used to situate the coordinate systems. In the next two sub-sections, we give a brief overview of these two. In this short paper, we only have space to describe enough of the construction process to provide a background for the example options in the next section.

B. *The Deixis Ontology - Situating the Coordinate System*

In practice, the local coordinate systems are centred and oriented around the platforms and sensors that use them. These platforms often have their centre and orientation physically marked using a series of physical plates specified in the design. We call this the object's deixis or *attitude*; its orientation in space-time. Typically, this deixis is conceptualised as three orthogonal axes (geometrically, lines) [23] – Fig. 1 above. We label these lateral, longitudinal and sagittal, using the standard anatomical deixis terms [24]. In the ontology, we separate this concern into a discrete deixis sub-ontology. This is then used as a basis for orienting the coordinate systems.

C. *The Coordinate System Ontology - Options*

Sets of co-oriented geometric surfaces are fundamental components of the ontology. In this section, we explain how these emerge and then look at the overall stages in the construction of the system.

The first two major options for a coordinate system are:

1. Reference frame – this is typically fixed by the object – the local reference frame.
2. Coordinate system's surface configuration type – in this case, one of the three surface configuration types in scope.

The three surface configuration types in scope are compositional: they each decompose into three reusable components – sets of co-oriented coordinate surfaces. What distinguishes the system types is the combination of different surface components, as enumerated in the TABLE I. below.

TABLE I. Coordinate System and their Surface Types

Coordinate System	Component Coordinate Surface Types
Cartesian	3 × planes
Spherical	sphere, cone and half-plane
Cylindrical	cylinder, half-plane and plane

The empirical evaluation of the different possible architectures against the data helped us to evolve a common, staged geometric construction process across the coordinate surface components - with variations for the different surfaces – as listed in TABLE II. below.

TABLE II. Common Staged Coordinate Surface Construction Process

Order	Stage	Description
1	Surface Orientation	Selecting the set of co-oriented surfaces
2	Solid Ordering	Building a mereological ordering for the surfaces – the process varies by surface.
3	Ratio Scaling	In these three systems, shifting down one or two dimensions to distance and angle ratios.
4	Unitising	Selecting the unitised distance or angle ratios – based upon the selection of unit.
5	Labelling	Labelling the unit ratios

As this shows, much of the work in constructing the overall system happens at the coordinate surface level. In this paper, we take most of our examples from the first two stages. The coordinate system is then assembled from its three surface components. The systems align the components, typically aligning their degenerate surface members, where these exist. The degenerate surfaces lose one or more dimensions, and so are lines or points – for example, the sphere with zero radius is a point. So, the intersection of three surfaces, one from each component set, picks out a point (as shown in Fig. 9) – and conversely, every point can be picked out by the intersection of three surfaces. So when each coordinate surface is given a coordinate numeral label, each point is named by the coordinate triple composed of the coordinate labels of the three surfaces that include it.

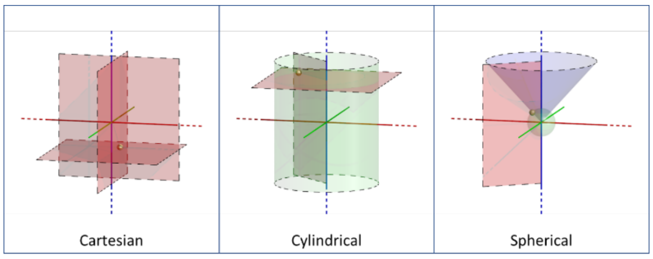

Fig. 9. Three Coordinate Systems - Showing the Intersecting Surfaces That Identify a Point

V. Options as Levels - Examples

The previous sections have set up the context for the coordinate system examples of options as levels. The coordinate system ontology's broad stages contain a series of steps, which sometimes involve options. We have selected a examples to illustrate the two types of options (combinations and permutations) and how these merge from the constructional ontology.

A. *Coordinate Surfaces Orientation Combinations*

As noted above, selecting the object selects the reference frame. And, selecting the type of coordinate system, picks out the types of the three surfaces that will be used. Each of the places in the coordinate triple contains a numeral that labels a surface of the chosen type.

To see what surfaces these are, consider the object at a point in time. Assume that the selected coordinate surface type is planes. Consider all the possible planes in its local reference frame. Partition these into sets of planes that are parallel to one another – the set of these is 'co-oriented plane types', a combination. Each of these subsets will contain planes that cover the whole of space; in other words, each point in space will be in one and only one plane in every subset. Also, all the subsets are disjoint, as members from different subsets will not be parallel – so cannot be in the same subset. A visualisation of this construction in Fig. 10 shows how generational levels underpin the 'co-oriented plane types' combination.

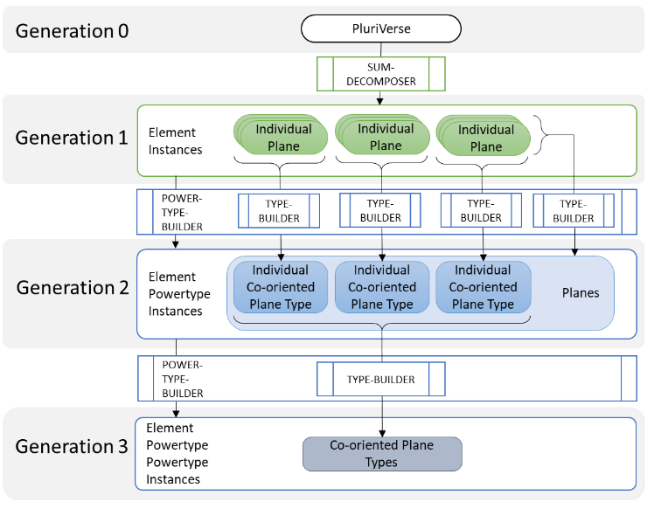

Fig. 10. Coordinate Plane Surface – Generational Levels

There is a similar geometric construction with variations for the other surfaces. For example, in the case of spheres, we start as before, by considering all the possible spheres in the objects' reference frame. We then partition these based on sharing a centre – in other words, being co-centred, hence co-oriented.

When setting up the coordinate surface for the coordinate system, the surface's orientation needs to be selected. In the case of planes, spheres and the other surfaces, the options for orientation are the instances of the relevant sets of co-oriented surfaces – which are level (generation) 3 objects – visualised in Fig. 11. The three surfaces can then be grouped into a coordinate proto-system – a system with only orientation.

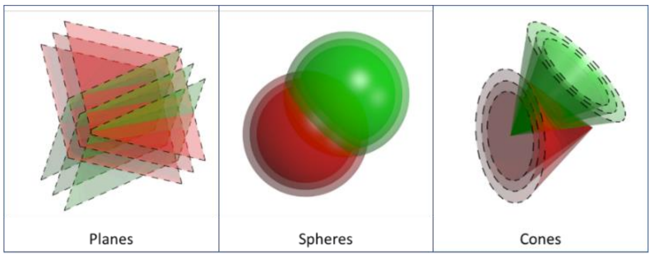

Fig. 11. Visualisation of Sets of Co-oriented Surfaces – Planes, Spheres and Cones.

B. *Conical Surface Solidification Combinations*

The goal of the solid ordering or solidification process in TABLE II. is to end up with a set of solids – one for each surface – where the solids are mereologically linearly ordered – so, for every solid each of the other solids is either a part of it or has it as a part. This is needed for the next ratio scaling phase, which uses anthyphairesis – a mereologically based process of reciprocal subtraction; for details see [25, 26].

The analysis shows that the surfaces require different solidification constructions with different levels of options. It took some investigation to devise elegant and parsimonious constructions for some of the more complicated constructions. Spheres and cylinders are relatively simple with no pragmatic options; the solid is their finite interior (it would be unnatural to select their infinite exteriors).

Cones are a useful example of something with a less simple, but not too complicated option. Co-orientation partitions the set of cones (surfaces) into disjoint sets of co-oriented cones. It similarly partitions the set of conical solids into corresponding disjoint sets of co-oriented conical solids; in other words, each set of co-oriented cones has one corresponding set of co-oriented conical solids. One way of visualising this is to consider how each cone in a set of co-oriented cones, being infinite, divides the space into two half-spaces, both of which are in the corresponding set of co-oriented conical solids – giving a one-to-two mapping, see Fig. 12.

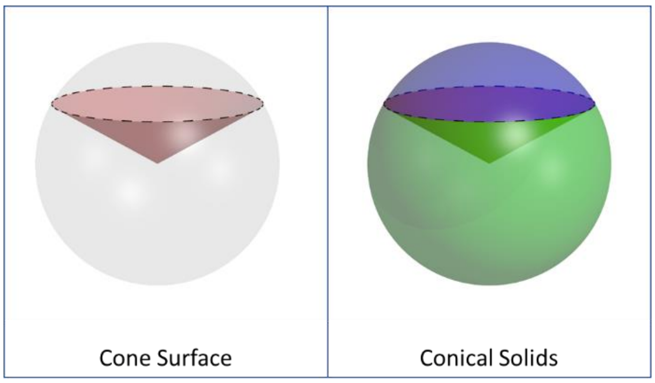

Fig. 12. Cone Dividing Space into Two Conic Solids

This one-to-two mapping from surfaces to solids is a specific case of a more general situation that is reflected downstream in ambiguity of angle identification. In the simplest case, in plane geometry, where an angle is defined as a relationship between two rays meeting at a vertex, there is an ambiguity about which of the two angles is intended (in our case, it is which of the two solids) – see Fig. 13 below. The rays by themselves are insufficient to distinguish between the two possible angles.

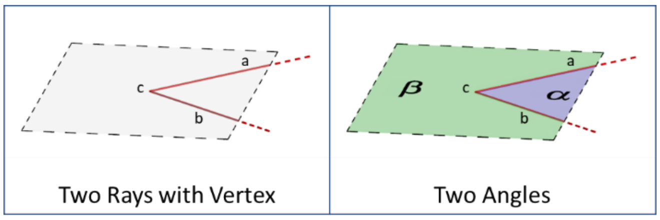

Fig. 13. Angle Identification

This can be avoided if one defines ‘angle’ in a similar way to the Ancient Greek Carpus of Antioch, as a space between the lines - quoted in [27, pp. 125–126]. Then in Fig. 13, there are two spaces, α and β, and therefore, two angles. Our solidification strategy moves up a dimension and takes the volume between two surfaces.

The Carpusian solid angle approach enables us to distinguish between the two angles. However, we also need a simple, consistent way to choose between the two conical solids (angles) associated with each cone in a set of co-oriented cones – which preserves a linear mereological ordering that is then reflected in the eventual labelling. We do this by exploiting the fact that each set of co-oriented conical solids contains two rays as degenerate solids: where the three-dimensional conical solids collapse into a one-dimensional ray (half-line) with no volume. These two solids are mereologically minimal; in other words, no other solid in the set is part of them. We then divide the solids sets into two subsets depending upon which of the degenerate solids they have as a part. We can use the degenerate solids as an index for each subset. Importantly, each subset is the basis of a different coordinate system component (hence a combination option), as it will result in a different way of labelling the solids (and so surfaces) with an angle. If we do not know which subset was chosen, we cannot interpret the label for the angle.

C. *Coordinate Surface – Planar Solid Sub-Sets Permutations*

The solidification process for planes is more complex, involving more options. As for cones, co-orientation partitions the set of planes (surfaces) into disjoint sets of co-oriented planes. In the case of planes, one constructs the set of parallel-boundaried planar solids – where one takes every pair of parallel planes (possibly identical) and constructs the solid that has the two planes as boundaries; in other words, the interior between the two planes (it would be unnatural to select either of the two single boundaried exteriors). Then co-orientation partitions this set into corresponding disjoint sets of co-oriented (parallel) planar solids; where each set of co-oriented planes has one corresponding set of co-oriented planar solids.

Note that every plane (surface) is contained in this parallel-boundaried planar solids type as a degenerate solid. Associated with each degenerate solid is a subset of this type containing every solid that has the degenerate one as a boundary. In coordinate system terms, these subsets represent the combination options for picking a reference plane, namely, the plane associated with the subset. The mereological structure of the solids in each subset induces two linear orderings each of which covers a half-space – the two orderings have the reference plane as their only common element. Intuitively, these orderings arrange mereologically the solids on each side of the reference plane.

The scaling-ratio-unit process will label the surfaces via the solids. But, the labels will not be unique as things stand – they will appear twice, once in each subset. So, the subsets need to be differentiated (rather than one selected) and there are two ways (in the sense of permutations) of doing this, each way giving a different labelling and thus a different coordinate system. One can do this with a couple, where for clarity, we label place 1 as positive and place 2 as negative. Hence, this is a permutation option.

D. *Deixis Mapping to Cartesian Proto Co-ordinate System Permutations*

As noted at the beginning of this section, we extract the object’s attitude (its orientation in space-time) from the coordinate system ontology into a deixis sub-ontology. This deixis is used to situate the coordinate system, and so the component coordinate surfaces. It turns out that the deixis axes are not fundamental to the pure coordinate system structure, and so are analysed away. However, they play a critical part in how the deixis situates the coordinate system. We illustrate this with an example that focuses on orientation, as this is both clearer and simpler than accounting for the full coordinate system.

We can define the orientation of a coordinate system as the set of orientations of its component surfaces. In our terms, the orientation of a coordinate surface is the set of co-oriented surfaces (this is analogous in some ways to Frege’s abstraction from parallel lines to directions [28]). This notion of oriented surfaces can be easily extended to account for the orientation of a coordinate system – and regarded as a stage in its construction. An oriented coordinate system is one where its three surfaces have all been oriented; we call this a proto coordinate system.

As a side note, from this perspective, the Cartesian coordinate system’s orientation does not involve axes directly – it is just a set of three co-oriented plane types. The conventional way of representing this with three orthonormal axes, one for each plane surface type, may be simpler to visualise but is misleading about the underlying ontology – as any of the infinite lines parallel to it will construct an identical orientation, the same set of co-oriented planes. One can regard the axes as the mechanisms for constructing the co-oriented surfaces rather than co-orientation itself.

However, the Cartesian axes are a good foundation for the deixis geometry – the object’s attitude – as these are not arbitrary as they go through the object’s centre. The three lines (axes without direction) are the deictic base orientation. However, some aspects of this orientation are abstracted away in the mapping to the Cartesian coordinate system. One way to appreciate this is to note that the same Cartesian proto system is constructed in the cases where the deictic axes are swapped (in other words, the axes are permuted). Physically, this would happen if an Unmanned Underwater Vehicle were to rotate in a way that any of its three deictic axes were swapped – as shown in Fig. 14.

The six configurations in Fig. 14 are the six ways that the situated object can view the Cartesian coordinate proto-system. Another way to look at it is as the six ways to order the three surfaces or as six ways to interpret the three Cartesian coordinates relative to the object’s attitude. Given three coordinate labels {x, y, z}, which is longitudinal (up-down) from the deictic perspective? Any one of the three could be. Hence the deictic base orientation has six permutations of the underlying Cartesian coordinate system orientation – in other words, this is a permutation option. For the full system, one would need to consider other factors, such as direction (for example, up versus down) which would multiply out the permutations.

In a single platform sensor system, there is less of a need to consider this point. There is less of a requirement to distinguish between the deixis and coordinate systems, and hence historically many single platform systems have merged the two.

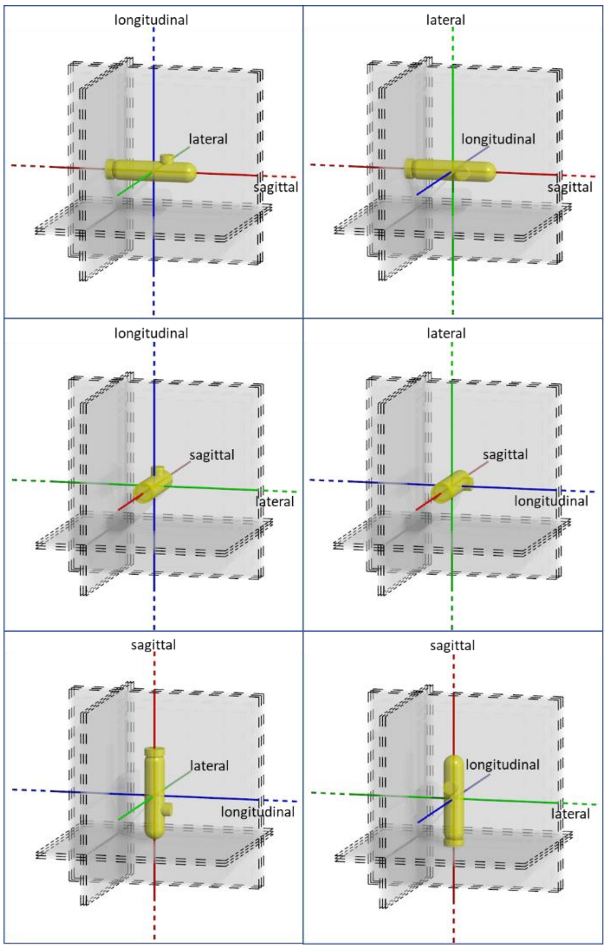

Fig. 14. Six Permutations of The Base Deictic Orientation.

VI. Conclusion

We have provided examples that show how both kinds of option (combination and permutation) can arise in enterprise projects. We have shown how the constructional approach can be used as an analytic tool to identify the foundation levels in a domain. And, how this approach gives a clear picture of how these options fundamentally involve ascending levels – and so are intrinsically multi-level. We have also shown, through the permutation examples, how tuples/relations multi-levelling complements type multi-levelling.

The unearthing of multi-levelling in a geometric domain should add weight to the claims (often made in the community) that multi-level modelling pervades conceptual models. The claim we have argued for (that options are inherently level ascending) should add further weight.

Though it is not the main goal of the paper, we have provided some insight into how the conceptual foundation for a multi-platform sensed position coordinate system could be developed using the constructional approach and what it would look like. We have included some examples that show how the underlying fundamental structure is not transparent. For example, how the analysis replaces the traditional visualisation of the Cartesian system as three axes with the less easy to visualise but more correct three co-oriented plane types. These help make a case for a foundational analysis that can reveal the underlying structure.

Finally, we have illustrated how a foundational approach can help to future-proof systems. Single platform-domain systems whose development focused on specific requirements without a conceptual model of the foundations are not designed with the options in mind and so cannot easily accommodate the move to multi-platform-domain.

Acknowledgment

We want to acknowledge the help Steven Bradley and Salvatore Florio gave us in the development of this paper.

References

[1] U. S. D. Defense, *Unmanned Systems Integrated Roadmap FY2017 - 2042*. (2018) [Online]. Available: https://www.hsdl.org/?abstract&did=826737.

[2] U. S. D. Defense, *Unmanned Systems Integrated Roadmap FY2011 - 2036*. (2011) [Online]. Available: https://www.hsdl.org/?abstract&did=705359.

[3] C. Partridge *et al.*, “Semantic Modernisation: Layering, Harvesting and Interoperability.” in NATO Symposium IST-101 / RSY-024, Semantic and Domain-based Interoperability. (2011).

[4] C. Partridge *et al.*, “Developing an Ontological Sandbox: Investigating Multi-Level Modelling’s Possible Metaphysical Structures.” in MULTI-4th International Workshop on Multi-Level Modelling. pp. 226–34 (2017).

[5] C. Partridge, *Business Objects: Re-Engineering for Re-Use*. Butterworth-Heinemann (1996).

[6] C. Partridge *et al.*, “Formalization of the Classification Pattern: Survey of Classification Modeling in Information Systems Engineering.” Software & Systems Modeling. pp. 1–37 (2016).

[7] C. Partridge *et al.*, “A Novel Ontological Approach to Semantic Interoperability Between Legacy Air Defence Command and Control Systems.” International Journal of Intelligent Defence Support Systems. vol. 4, pp. 232–62 (2011).

[8] M. Lambert *et al.*, “Demonstrating a Successful Strategy for Network Enabled Capability.” in NATO Symposium IST-101 / RSY-024, Semantic and Domain-based Interoperability. (2011).

[9] OMG, “Open Architecture Radar Interface Standard (OARIS).” (2016).

[10] ISO, “ISO/IEC 18026: 2009 - Information Technology - Spatial Reference Model (SRM).” (2009).

[11] P. Suppes *et al.*, “Foundations of Measurement, Vol. II: Geometrical, Threshold, and Probabilistic Representations.” (1989).

[12] F. Arntzenius *et al.*, “Calculus as Geometry.” in Space, Time, and Stuff. Oxford University Press (UK) (2014).

[13] I. Grattan-Guinness, “Numbers, Magnitudes, Ratios, and Proportions in Euclid’s Elements: How Did He Handle Them?” Historia mathematica. vol. 23, pp. 355–75 (1996).

[14] C. Partridge, “Geospatial and Temporal Reference - A Case Study Illustrating (Radical) Refactoring.” in ONTOBRAS-2013 6th Ontology Research Seminar in Brazil. (2013) [Online]. Available: https://www.academia.edu/27433806/Geospatial_and_temporal_reference_A_case_study_illustrating_radical_refactoring.

[15] C. Partridge, “An Information Model for Geospatial and Temporal Reference.” (2011) [Online]. Available: https://www.academia.edu/39988229/An_Information_Model_for_Geospatial_and_Temporal_References.

[16] S. de Cesare *et al.*, “BORO as a Foundation to Enterprise Ontology.” Journal of Information Systems. vol. 30, pp. 83–112 (2016).

[17] C. Partridge, “Note: A Couple of Meta-Ontological Choices for Ontological Architectures.” LADSEB CNR, Padova, Italy. (2002).

[18] K. Fine, “The Study of Ontology.” Noûs. vol. 25, pp. 263–94 (1991).

[19] K. Fine, “Towards a Theory of Part.” The Journal of Philosophy. vol. 107, pp. 559–89 (2010).

[20] J. Schaffer, “Monism: The Priority of the Whole.” Philosophical Review. vol. 119, pp. 31–76 (2010).

[21] B. Russell, “Logical Atomism.” in Contemporary British Philosophy. Personal Statements. pp. 356–83. ed. J. H. Muirhead. Allen & Unwin, London (1924).

[22] R. Carnap, *The Logical Structure of the World and Pseudoproblems in Philosophy*. tran. R. A. George University of California Press (1967).

[23] ISO, *ISO 1503:2008 - Spatial Orientation and Direction of Movement - Ergonomic Requirements*. (2008).

[24] L. H. Hyman, *Hyman’s Comparative Vertebrate Anatomy*. University of Chicago Press (1992).

[25] D. H. Fowler, *The Mathematics of Plato’s Academy: A New Reconstruction*. Clarendon Press Oxford (1987).

[26] Z. R. Perry, “Mereology and Metricality.” Forthcoming.

[27] Proclus, *A Commentary on the First Book of Euclid’s Elements*. tran. G. R. Morrow Princeton University Press (1970).

[28] G. Frege, *The Foundations of Arithmetic: A Logico-Mathematical Enquiry Into the Concept of Number*. tran. J. L. Austin Basil Blackwell & Mott (1950).