Higher-Order and Symbolic Computation, a Springer Journal

An Emerging Domain Science

A Rôle for Stanisław Leśhniewski's Mereology and Bertrand Russell's Philosophy of Logical Atomism

Dines Bjørner

Compile date: July 13, 2009 / Received:July 13, 2009 / Accepted: date

Abstract Domain engineers describe universes of discourse such as *bookkeeping, the financial service industry, container shipping lines, logistics, oil pipelines, railways*, etc. In doing so domain engineers have to decide on such issues as identification of that which is to be described; which of the describable phenomena and concepts are (to be described as) entities, operations, events, and which as behaviours; which entities are (to be described as) continuous which are (...) discrete, which are (...) atomic, which are (...) composite and what are the attributes of either and the mereology of composite entities, i.e., the way in which they are put together from sub-entities. For each of these issues and their composite presentation the domain engineer has to decide on levels of abstraction, what to include and what to exclude.

In doing so the domain engineer thus has to have a firm grasp on the a robust understanding and practice of the very many issues of description: *what can be described*, *identifying what is to be described*, *how to describe*, *description principles*, *description techniques*, *description tools* and *laws of description*. This paper will outline the issues in the *slanted type font*.

One issue, 'mereology', was studied, in a wider setting, by the Polish philosopher and mathematician Stanisław Leśhniewski, in the 1920s [55,66]. We shall relate our work to that of Leśhniewski and those of his followers who have suggested axiomatisations of various forms of mereology [53,23,52,24,22].

Similar issues, it appears, were studied, also in a wider setting, by Bertrand Russel in his lectures in 1918 and published in the Monist Journal, vols. XXVIII-XXIX in 1918-1919 [63]. In this essay we shall sketch how we see the rôle of "applied logical atomism" in domain engineering.

Keywords Domain Engineeering · Domain Science · Logical Atomism · Mereology

Invited paper for special issue of Journal of Higher-Order and Symbolic Computation, dedicated to the memory of Peter Landin 1930 – June 3, 2009

Dines Bjørner
Fredsvej 11, DK-2840 Holte, Denmark
Tel.: +45-45422141
E-mail: bjorner@gmail.com – URL: www.imm.dtu.dk/~db

1 Introduction

This paper may seem to "waver" between computing science and philosophy. In this introduction we will assume that the reader has read the abstract and then mention two strands of philosophical discourse cum mathematical logic around which this paper revolves.

1.1 On Trying to Understand The World

We focus on two strands of discourse: mereology and Bertrand Russell's Philosophy of Logical Atomism.[1]

1.1.1 Mereology

From ancient time and, perhaps, culminating in the 20th Century, thinkers have speculated on the logical structure of the world. Aristotle [3, Metaphysics, Book IV, Chapter 2] was perhaps the first person to consider the part-whole relationship. But it was not until the 1920s that the Polish mathematician Stanisław Leśniewski gave a formal treatment [54]. Woodger [77] and Tarski [68] made use of a specific adaptation of Leśniewski's work as a basis for a formal theory of physical things and their parts. The term 'calculus of individuals' was introduced by Leonard and Goodman [53] in their presentation of a system very similar to Tarski's adaptation of Leśniewski's mereology. Slightly earlier than Leśniewski's development of his mereology, Whitehead [74] and [73, The Concept of Nature] was developing a theory of extensive abstraction This system, according to Russell [60], was to have been the fourth volume of their Principia Mathematica, the never-published volume on geometry. Both Leśniewski [54] and Tarski [69, Foundations of the geometry of solids] have recognized the similarities between Whitehead's early work and Leśniewski's Mereology. Nelson Goodman mediated the Calculus of Individuals in [31, The Structure of Appearance]. Goodman had earlier used the Calculus of Individuals in his Ph.D. dissertation thesis [32, A Study of Qualities, 1940]. As in his joint paper with Leonard, he used the calculus as an addition to set theory to solve a problem known as the 'difficulty of imperfect community' (Rudolf Carnaps [21, Der Logische Aufbau der Welt]).

The field of mereology is still active, now part, mostly of computer science. Active researchers are Barry Smith [65, Mereotopology: A Theory of Parts and Boundaries], Achille C. Varzi [72, Spatial Reasoning in a Holey World[2]] and [71, On the Boundary between Mereology and Topology], and others.

1.1.2 Logical Atomism

Along a different line Bertrand Russell [63, Philosophy of Logical Atomism] (and [64, Vol. 8, Part III, Chap. 17, pp 157–244]) expressed a *metaphysics* which we summarise as:

[1] We spell mereology with lower case first letter as mereology is a concept that is not unique to its researchers whereas Bertrand Russell's Philosophy of Logical Atomism is.

[2] holey: something full of holes

> *the world consists of a plurality of independently existing things exhibiting qualities and standing in relations; all truths are ultimately dependent upon a layer of facts and these facts consist of either a simple particular exhibiting a quality, or multiple simple particulars standing in a relation.*

Russell also expressed a *methodology* for doing philosophy:[3]:

> *The methodology consists of a two phase process.*
> *The first phase is dubbed the "analytic" phase (although it should be noted that sometimes Russell used the phrase "analysis" for the whole procedure). One begins with a certain theory, doctrine or collection of beliefs which is taken to be more or less correct, but is taken to be in certain regards vague, imprecise, disunified, overly complex or in some other way confused or puzzling. The aim in the first phase is to work backwards from these beliefs, taken as a kind of "data", to a certain minimal stock of undefined concepts and general principles which might be thought to underlie the original body of knowledge.*
> *The second phase, which Russell described as the "constructive" or "synthetic" phase, consists in rebuilding or reconstructing the original body of knowledge in terms of the results of the first phase. More specifically, in the synthetic phase, one defines those elements of the original conceptual framework and vocabulary of the discipline in terms of the "minimum vocabulary" identified in the first phrase, and derives or deduces the main tenets of the original theory from the basic principles or general truths one arrives at after analysis.*

Together the two, the metaphysics and the methodology expresses an essence of Russell's 'Philosophy of Logical Atomism'. We shall, in this paper, relate this philosophy to domain science.

• • •

We shall examine our model (Sect. 6.2) of mereology in the light of proposed axiom systems for mereology Sects. 6.3–6.4, as well as with respect to Russell's Philosophy of Logical Atomism, Sect. 7. In addition to discussing Russell's ideas in Sect. 7, we shall repeatedly be referring to those ideas, especially in Sect. 5.

1.2 The Triptych Dogma

The present topic crystallized during the writing of [14]. During the writing of [5–7] I treated the problem of *"what we can describe"*[4] in a less than, to me, fully satisfactory way. In now writing my possibly last book [14], if ever accepted by the publisher, I had better do a more thorough job of understanding the issues now dealt with in this paper.

The background for our inquiry is that of the triptych dogma for software engineering: *"software cannot be designed without a robust understanding of the requirements"*; and *"requirements cannot be prescribed without a robust understanding of the*

[3] http://plato.stanford.edu/entries/logical-atomism/

[4] [what we can describe] of phenomena and concepts in at least those universes of discourse whose description, via requirements, eventually lead to computing

domain.”[5] Software *designs* are *specified*, *requirements* are *prescribed* and *domains* are *described*. Michael Jackson expresses this as follows: descriptions are *indicative*, prescriptions are *optative* and designs are *imperative* [39].[6]

The emphasis in this paper is on descriptions. The issues are *what can be described*, *identifying what is to be described*, *how to describe*, *description principles*, *description techniques*, *description tools* and *laws of description*. Since, as we show elsewhere, [7,10,14], requirements prescriptions can be readily “derived” from domain descriptions, and designs developed through requirements refinement, our study of these issues “carry over” to prescriptions and designs.

In ideal software engineering, one that is hardly attainable, one would therefore proceed, linearly, from *domain engineering* via *requirements engineering* to *software development*. In actual practice, professional software development proceeds through iterations between these three phases, forward and backward. This has been well illustrated in Michael Jackson’s *Problem Frame* approach [40].

1.3 Structure of Paper

Section 2 illustrates a domain description. It is that of simple (road, rail, shipping and/or air-lane) networks of hubs (road intersections, train stations, harbours, airport) and links (road segments, tracks between train stations, shipping lanes, air lanes). We illustrate only two facets, simple entities and operations, leaving out events and behaviours without loss of generality.

Section 3 sketches an answer to the question *“what are domains”* — with the rest of the paper more-or-less completing that answer.

In Sect. 4 we briefly survey the phases, stages and steps of domain engineering, some administrative, some technical/clerical, some engineering and some clearly bordering between engineering and research. That section thus provides a capsule overview of ‘domain engineering’.

Section 5 then focuses on *“what can be described”* and *“how do we describe”*. The section expresses a number of dogmas. These are referred back to Bertrand Russell’s thoughts on *“description theory”* [61,62,75].

Section 6 outline one aspect of ‘descriptions’: that of dealing with parts and wholes, including relations between parts and parts to the whole. This subject is called ‘mereology’. It first received its current forms of axiomatic treatment through the works of Stanisław Leśniewski [55,66].

Section 7 reviews Sect. 5 in light of Bertrand Russell’s *Philosophy of Logical Atomism* [63].

Finally Sect. 8 overviews a number of *laws of description*.

Section 9 concludes the paper.

2 An Example Domain Description

1. There are hubs and links.

[5] I thank Dino Mandrioli for his rewording of my: *“before software can be developed we must understand the requirements; before requirements can be prescribed we must understand the domain.”*.

[6] Michael Jackson, especially in his delight- and thoughtful [39], comes very close to several of the issues of the present paper.

2. There are nets, and a net consists of a set of two or more hubs and one or more links.

type
  1 H, L,
  2 N = H-**set** × L-**set**
axiom
  2 ∀ (hs,ls):N • **card** hs≥2 ∧ **card** ks≥1

3. There are hub and link identifiers.
4. Each hub (and each link) has an own, unique hub (respectively link) identifiers (which can be observed from the hub [respectively link]).

type
  3 HI, LI
value
  4a obs_HI: H → HI, obs_LI: L → LI
axiom
  4b ∀ h,h′:H, l,l′:L • h≠h′ ⇒
      obs_HI(h)≠obs_HI(h′) ∧ l≠l′⇒obs_LI(l)≠obs_LI(l′)

In order to model the physical (i.e., domain) fact that links are delimited by two hubs and that one or more links emanate from and are, at the same time incident upon a hub, we express the following:

5. From any link of a net one can observe the two hubs to which the link is connected.
   (a) We take this 'observing' to mean the following: From any link of a net one can observe the two distinct identifiers of these hubs.
6. From any hub of a net one can observe the one or more links to which are connected to the hub.
   (a) Again: by observing their distinct link identifiers.
7. Extending Item 5: the observed hub identifiers must be identifiers of hubs of the net to which the link belongs.
8. Extending Item 6: the observed link identifiers must be identifiers of links of the net to which the hub belongs

We used, above, the concept of 'identifiers of hubs' and 'identifiers of links' of nets. We define, below, functions (iohs, iols) which calculate these sets.

value
  5a obs_HIs: L → HI-**set**,
  6a obs_LIs: H → LI-**set**,
axiom
  5b ∀ l:L • **card** obs_HIs(l)=2 ∧
  6b ∀ h:H • **card** obs_LIs(h)≥1 ∧
  ∀ (hs,ls):N •
  5a   ∀ h:H • h ∈ hs ⇒ ∀ li:LI • li ∈ obs_LIs(h) ⇒
        ∃ l′:L • l′ ∈ ls ∧ li=obs_LI(l′) ∧ obs_HI(h) ∈ obs_HIs(l′) ∧
  6a   ∀ l:L • l ∈ ls ⇒
        ∃ h′,h″:H • {h′,h″}⊆hs ∧ obs_HIs(l)={obs_HI(h′),obs_HI(h″)}

7 $\forall$ h:H • h $\in$ hs $\Rightarrow$ obs_LIs(h) $\subseteq$ iols(ls)
8 $\forall$ l:L • l $\in$ ls $\Rightarrow$ obs_HIs(h) $\subseteq$ iohs(hs)
value
iohs: H-**set** $\rightarrow$ HI-**set**, iols: L-**set** $\rightarrow$ LI-**set**
iohs(hs) $\equiv$ {obs_HI(h)|h:H•h $\in$ hs}
iols(ls) $\equiv$ {obs_LI(l)|l:L•l $\in$ ls}

In the above example we have focused on just five entities: nets, hubs, links and their identifiers. The nets, hubs and links can be seen as separable phenomena. The hub and link identifiers are conceptual models of the fact that hubs and links are connected — so the identifiers are abstract models of 'connection', or, as we shall later discuss it, the mereology of nets, that is, of how nets are composed. These identifiers are attributes of entities.

We have not mentioned other attributes of hubs and links — such as their 'location', 'name', 'length' of links, 'modality' of hubs and links, i.e., whether of road, rail, ship or aircraft modality, etc. And we have not mentioned attributes of nets — such as 'in which country a net is located', 'which authority has responsibility for the net', 'whether a road link is a freeway, a toll road, or other', etc.

3 What Are Domains ?

When, in this paper and elsewhere, we speak of domains, we speak primarily of man-made universes of discourse, interacting with the universe such as studied by physicists (including chemical scientists). We listed a few references to 'domains' in the first sentence of the abstract. We list two more here, and, in parentheses we list some representative simple entities, operations and events: the consumer market (consumers, retailers, wholesalers, producers, distribution chain, goods, catalogues, etc.; inquire, order, accept, payment, reject, return; erroneous shipment, failure of delivery; failure of payment), and hospital health care (patients, medical staff, wards, beds, hospitalisation plans, patient medical records; anamnese, analysis, diagnostics, treatment; declared cured, declared dead).

3.1 An Attempt at A Characterisation: Domain Science versus Physics

Domain descriptions, such as we shall advocate them, emphasise the discrete logical structure, properties and operations of the domain. Usually the number of observed types of entities is large — of the order of up to several 100s. This is in contrast to physics descriptions which appear to emphasise the the continuous behaviour of a few — 5-10 types of — state variables.

Thus, what appears to characterise domains, say in contrast to physics, is the immediate algebraic nature of the universe of phenomena and concepts of domains. So a first mathematical cum computing science characterisation of domains focuses on their being heterogeneous algebras: a finite set of indefinite sets of simple entities (particulars) of different types and a finite set of operations whose signatures range over entity types, including higher-order functions over these. Included in this first characterisation is that the simple entities are usually discrete and either atomic or composite and otherwise possess observable attributes.

This is in contrast to the entities and operations of physics: usually infinite, usually continuous, and seen (i.e., modelled) more as state variables satisfying differential and integral equations and possessing statistical and probabilistic properties and otherwise being representable in terms of real numbers of the types that can be derived from or are directly expressed in terms of length, mass, time, electric current, thermodynamic temperature, amount of substance and luminous intensity. Physics is the study of matter and its motion through space-time and all that derives from these, such as energy and force. More broadly, it is the general analysis of nature, conducted in order to understand how the world and universe behave.

In physics (as characterised above) the physicists, in principle, do not include human actions and behaviours in their study.[7]

Domain scientists and domain engineers — the borderline between when one is a domain scientists and one is a domain engineer is, at present, somewhat vague — study the structures (entities) conceived and built by humans (the domain owners, managers, designers, domain enterprise workers, etc.), and the operations that are initially requested, or triggered, by humans (the domain users). More characterisations follow below.

3.2 Tools of Domain Science & Engineering

The tools of domain scientists and engineers are those of careful, precise informal (i.e., narrative) natural language and likewise careful abstractions expressed in some formal specification languages emphasising the algebraic and logical nature of entities and their operations. That is, tools that originate with computer and computing scientists.

Why not use the same tools as physicists do? Well, they are simply not suited for the problems at hand. Firstly the states of physics typically vary continuously, whereas those of domains typically vary in discrete steps. Secondly the number of state variables of physics do usually not vary, whereas those of domain do — whole structures "collapse" or "expand" (sometimes "wholesale", sometimes "en detail"). Thirdly the models of physics, by comparison to those of domains, contain "only a few types" of oftentimes thousands of state variables — almost all modelled as reals, or vectors, matrices, tensors, etc., of reals, whereas those of domains contain very many, quite different types — sometimes atomic, sometimes composite, but rarely modelled (abstractly) in matrix form.

Models of physics, as already mentioned, express continuous phenomena. Models of domains, as also already mentioned, express logic properties of algebraic structures.

For those and several other reasons the tools of physicists are quite different from the tools of domain scientists and engineers.

In theoretical physics there is no real concern for computability. Mathematical models themselves provide the answers. For domain engineering there is a real concern for computability. The mathematical domain models often serve as a basis for requirements for software, that is, for computing. Hence it was natural that the tools of domain science and engineering originated with the formal specification languages that were and are used for specifying software.

[7] The claimed possibility that humans are the origin, through their use of fossil energy sources, of the depletion of the ozone layer, does not mean that the physicists, in their model include human actions and behaviours: if physicists do consider humans as the "culprits", then that still does not enter into their models !

4 Domain Engineering

There are several stages to this initial phase of software development. These stages are all carefully covered in [7, 14]. Here we give but a condensed review. The purpose of bringing this material is to point out all those stages, sub-stages and steps in the development of a domain description in which descriptional issues occur. That is, all those myriad of situations throughout the development process in which the software (cum domain) engineer is expected to have a robust understanding and practice of the very many issues of description: *what can be described*, *identifying what is to be described*, *how to describe*, *description principles*, *description techniques*, *description tools* and *laws of description*.

⋆ Information:[8] Informative documents must be developed in which practical information on the domain engineering project of developing a domain description is given: project name; project partners (names, addresses, persons, etc.); the situation in which the project takes place (problems of current understanding of the domain); which need is there for a domain description; which ideas will a domain description be based on; which are the main phenomena and concepts of the domain that will be studied; what is the (wider) scope and the (narrower) span of the domain being investigated; a project plan (budget, financing, resource plan, etc.); standards compliance; contracts and design briefs; logbook — etcetera. Although these documents are not explicitly describing, but at most "mentioning" domain phenomena and concepts, several description issues do apply to their formulation.

Stakeholder Identification: Identification of and ongoing liaison with as wide a range of domain stakeholders as possible: anyone with some engagement in the domain, from "owners", via managers, workers, users, suppliers, regulatory bodies, etc. must be contacted and interacted with on a regular basis — as seen in the following.

⋆ Domain Acquisition: Knowledge about the domain, as see "through" the "eyes, hands and brains" of each class of stakeholders must be gathered.

⋆ Business Process: Often domain acquisition can additionally be structured around the rough sketching of some of the business processes of the domain.

⋆ Domain Analysis: What has been gathered of knowledge (and represented through the domain acquisition units and business process sketches) must be analysed with respect to (relative) completeness, inconsistencies and "factual" contradictions[9]. Domain analysis is a first serious stage of domain model development in which basic issues such as studied by Bertrand Russell are manifested. We shall treat these throughout and summarise them in Sect. **??**.

⋆ Terminology: Throughout a terminology must be established, clear definitions given, adhered to, used and referred to in all relevant contexts and updated.

⋆ Domain Description: This stage could be seen as the crucial stage of domain engineering — the stage for which all the preceding stages prepare the domain engineer. We shall comment on this stage below and in Sect. 5 in detail.

Domain Description Verification: During the development of the domain description, and in its descriptional pre-stages: acquisition, analysis and terminology, the domain engineer, when formalising the domain description, will need to formally verify, model check and formally test tentative model descriptions.

---

[7] The issues of this paper is of relevance to items marked ⋆.

[9] Domain acquisition unit contradictions: Different stakeholder groups insist on the validity of contradicting presentations of the domain such as they see it

Domain Description Validation: All contacted and inquired stakeholder groups must "sign off" on the informal domain description, that is, the narrative.

⋆ Domain Theory: The ultimate objective of having a domain description is that it forms the core of a domain theory, a basis on which to study and develop a number of propositions, lemmas, theorems and laws about the domain model — and hence, in an informal sense, the domain.

There are several sub-stages of most of the above stages. Two of immediate interest to our study are 'Domain Acquisition' and 'Domain Description':

Domain Acquisition: We briefly look at the most important sub-stages of domain acquisition.

- Sources: First one must identify, besides the stakeholders, "independent", written sources of knowledge about the domain being inquired: books, journal articles, reports, the Internet (Wikipedia, etc.), etcetera. Equipped with what has been ascertained though these sources, the domain engineer is better prepared to meet the stakeholders.
- Contacts: Personal contact must be made, on a regular basis, with each and every stakeholder group. They must be privileged and authorised to follow any and all emerging domain engineering documents.
- Interviews: Interviews serve to ensure that domain engineers and stakeholder do not "speak past one another", i.e., have their "spinal chords harmonised" !
- ⋆ Questionnaires: From studying "independent" sources, from contacts and interviews the domain engineer may have to formulate and explain domain acquisition questionnaires for stakeholders to fill out.
- ⋆ Domain Description Unit Handling: These questionnaires lead to the itemisation, annotation, multiple category classification and database registration of usually a very large set of domain description units for purposes of study during the domain analysis, terminology and description stages.

⋆ Domain Description: A domain description describes the inquired domain **as it is**, not as the stakeholders and domain engineers would like it to be, not as requirements to some computing & communication system, and certainly not in the form of directives as concerns any software designs. There are several ways in which the domain engineer can examine a domain. We list the most crucial such facets.

- ⋆ Domain Intrinsics: Initially one starts describing the very basic phenomena and concepts of the domain — those on which the description of any of the following facets must be based.
- ⋆ Domain Support Technologies: Often concepts — being defined in terms of described phenomena (and other such concepts) — are abstract and hide crucial "implementation" details. A support technology is then either a human or a hard engineering gadget which represents these concepts. **Example:** A railway switch, as a concept, is an entity that can take on either of a number of states (typically 12), each modelling one or more of the paths of trains through the switch: from the one end to the other, or vice versa, either along the straight path of the switch, or the curved ("deviation") path. Such an intrinsic model may be 'implemented', as in "ye olde days", by a railway worker throwing a heavy counterweight and pulling or drawing the switch, or by mechanical wires running over pullers etcetera to a cabin tower, or by electro-mechanical renditions of this, or by combining, as in modern day interlocks, the functioning of

groups of switches by electronic and electro-mechanical means. □[10] Support technologies must be described, not only when functioning correctly, but also when failing to do so.

- ⋆ Domain Rules & Regulations: Domain rules describe how support technology and humans are expected to behave in the domain. When it has been detected that a domain rule has been "disobeyed", a domain regulation stipulates how a possibly erroneous situation might be "brought ajour". Rules are like predicates: they apply to pairs of states: pre- and post-states of actions. Regulations are like actions to be applied in case a rule has been flaunted — with the action bringing the domain into a state from which it is meaningful to proceed.
- ⋆ Domain Scripts: Scripts are like sets of rules & regulations, like "programs" to be "carried out", usually by humans, i.e., stakeholders of the domain.
- ⋆ Domain Management & Organisation: Managers set strategic, tactical and operational goals, see that there is a budget for achieving these, allocating and scheduling resources, incl. people, to carry out these, and, as part of that, formulating and enforcing rules & regulations and scripts. Organisations structure management and workers at all levels, functionally, administratively and location-wise (possibly geographically widely dispersed).
- ⋆ Domain Human Behaviour: Some stakeholders behave "nicely", as positively expected (by other stakeholders), others sometime behaves sloppily, yet others more systematically behave in a delinquent manner, and finally there are those who behave outright criminally. This "smooth" spectrum of behaviours must be described. In addition one must describe the knowledge that humans may have with respect to their tasks (and their expertise in carrying out these tasks) in the domain: what they might know and believe (about other agents), promise and commit (to other agents), what is necessary, probable and/or possible, etcetera.

A domain description is then presented as a suitable "blend" of the above, not necessarily with the above sub-stages each resulting in separate description parts.

[7, Part IV, Chaps. 8–16, pp 193–362] and the book [13] give many examples of domain specifications. So does [14, draft textbook, submitted for evaluation].

5 On Domain Descriptions

In this section we shall cover the following description issues: *what can be described*, *how to describe*, *description principles*, *description techniques* and *description tools*.

5.1 On Designation of Particulars

How do we get started on a description and how do we proceed ?

We refer to Sect. 1.1.2 on page 2 whose indented paragraphs on 'metaphysics' and 'methodology' strongly hints at our approach.

The 'metaphysics' paragraph suggests that one starts with atomic entities. We mostly choose to start with atomic simple entities, but sometimes we start by first

[10] □ designates 'end of example'.

identifying atomic, or, as we shall call them, primitive operations. The 'methodology' paragraph tells us to first analyse these atomic entities and then their compositions.

The above approach shall determine our approach to domain description. An emphasis will then be to identify and describe the plurality of independently existing things, their qualities and how they relate to other things.

In Sect. 2 we only recorded a result of a synthetic phase.

The domain engineering stages of 'Domain Acquisition', 'Business Process Sketching', and 'Domain Analysis' exemplify analytic stages and the domain engineering stages of 'Terminoligisation' and 'Domain Description' exemplify synthetic stages.

5.2 Narrative and Formalisation, Domain and Model

In different contexts, that is, for different aspects of a domain to be described, we may use one or another such formal notation, i.e., formal specification language, even plain mathematics[11] In this paper we shall be using the `RAISE` specification language `RSL` [30,5–7,29]. For large fragments domains we could as well use `Alloy` [38], `B` [1, 20], `Event B` [2], `VDM-SL` [17,18,28,27]] or `Z` [76,34]. For other domain fragments we may *additionally* need to deploy `Duration Calculus (DC)` [78,79], `Message Sequence Charts (MSC)` [35–37], `Petri Nets` [57–59], `Statecharts` [33] and/or `Temporal Logic of Actions (TLA+)` [41,56].

[16] covers several of these and some other formal specification languages.

Based on the 'resolution' of Sect. **??** we can now make precise what we mean *syntactically* by a 'description'.

- To us a 'description' is presented as a sequence of pairs of related *narratives* and *formalisations*.
- Each formalisation either defines a class (i.e., type) of simple entities, or a function (including a behaviour), or a communication medium between behaviours (as well as the type of channel events).
- The simple entities, functions, events and behaviours portray phenomena observable in the domain or concepts formed from such.

We illustrated such a sequence in Sect. 2. The narrative is a precise natural language presentation of facts about a domain.

Semantically a formal, mathematical description

- denotes whatever is prescribed by the semantics of the formal specification being used, usually this is some mathematical structure;
- in the case of `RSL` it is a set of algebras, i.e., mappings from type, function, etc. names to mathematical structures as constrained by the axioms.

Pragmatically an informal, narrative description

- designates, for each of its named simple entities, functions, events and behaviours a corresponding phenomena, "out there", in the actual domain, while
- the relation between the informal narrative names and the real, designate phenomena, is informal.

[11] A formal specification language has a precise syntax, a precise semantics and a more-or-less comprehensive set of proof rules. Mathematics does not.

The intended relationship between informal text parts and the mathematical formula parts must therefore be kept as "obvious" as possible so that we can have a high trust in the mathematical model of the formalisation.

5.3 What Is It That We Describe ?

Instead of using the term 'particulars' we shall, in the following, use the term 'entities'. So we describe entities, that is: phenomena and concepts.

5.3.1 Time/Space Facts and Universals

We consider domains, that is, predominantly human-made universes of discourse, to exemplify a wide variety of independent and time/space phenomena as well as what Russell calls *universals* (we shall soon characterise what is meant by 'universals'). For us, a domain phenomenon exists spatially if it can, at any one time of its existence, be located in space. For us, a domain phenomenon exists temporally if it can be "pin-pointed" in time (and hence space). No two distinct, independent and time/space phenomena can occupy the same spatial location at any one time. Let us call the independent and time/space phenomena by the name 'time/space facts'.

In addition to time/space facts there are the (non-time/space fact) domain universals.

5.3.2 Concepts

And there are concepts and these are defined in terms of time/space and universal facts and other concepts. Concepts are abstractions and, we decide, concepts may have time/space relations. Such concepts would typically be simple, rather immediate abstractions of time/space and/or universal facts.

5.3.3 An Ontology of Phenomena and Concepts

These phenomena (facts) and concepts may, for practical, i.e., ease of mastering descriptional complexity, be considered entities of either of four kinds: *simple entities*, *operations* (over entities), *events* (involving change of some entity values), and *behaviours* — the latter as sets of sequences of possibly coordinated (synchronised, timed) operations and events.

5.3.4 Simple Entities

We shall just consider 'simple entities'. We use the name 'simple entities' in contrast to 'entities' which we see as comprising all simple entities, functions, events and behaviours. "Interesting" functions and normal events involve all forms of entities.

Simple entities are, as mentioned: "separately existing things". These "things" exhibit qualities, i.e., possess values, or more precisely, have a number of one or more attributes (of type and value) and zero, one or more sub-entities, the totality of which can be said to exhibit a simple entity value. These "things" can also relate to other things such as being part of another simple entity or being "together" with other similar things, i.e., standing in some relation to other "things".

Continuous and Discrete Simple Entities: A simple entity is either continuous or is discrete, and then it is either atomic or composite.

A simple entity is said to be continuous if it can be arbitrarily decomposed into smaller parts each of which still remain simple continuous entities of the same simple entity kind. In this paper we shall not further consider continuous entities.

A simple entity is said to be discrete if its immediate structure is not continuous.[12] A simple discrete entity may, however, contain continuous sub-entities.[13]

By an attribute we mean a property of an entity. Typically we express attributes by a pair of a type designator: *the attribute is of type* V, and a value: *the attribute has value* v (of type V, i.e., $v : V$). A simple entity may have many properties.

Simple Entities versus Attributes: We make a distinction between simple entities and attributes. Simple entities are phenomena or concepts that have separate existence; that may be separate parts of other simple entities; that may (thus) be composed into other (composite) simple entities; and that otherwise possess one or more attributes. Attributes are properties of simple entity phenomena and concepts that together form atomic simple entities, or characterise composite entities apart, i.e., in isolation from their sub-entities; that cannot be "removed" from a simple entity otherwise possessing such attributes; and that may be modelled as values of simple or composite types.

When, above, we spoke of 'universals', we now speak of attributes, that is, Russell's notion of 'universal' is equated with our use of 'attribute'.

A Spatial Phenomenon: Some phenomena (p:P) enjoy the meta-linguistic "$\mathcal{L}$ property", $\mathcal{L}$ for $\mathcal{L}$ocation. Let any phenomenon and its derived concepts be subject to the meta-linguistic predicate, has_$\mathcal{L}$, and function, obs_$\mathcal{L}$:

type
  P, C, $\mathcal{L}$, E = P|C
value
  $\emptyset$:$\mathcal{L}$
  has_$\mathcal{L}$: E $\rightarrow$ **Bool**
  obs_$\mathcal{L}$: E $\stackrel{\sim}{\rightarrow}$ $\mathcal{L}$, **pre** obs_$\mathcal{L}$(e): has_$\mathcal{L}$(e)
  $=,\neq$: $\mathcal{L} \times \mathcal{L} \rightarrow$ **Bool**
  $\sqcap,\sqcup$: $\mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$
  $\sqsubset,\sqsubseteq$: $\mathcal{L} \times \mathcal{L} \rightarrow$ **Bool**
axiom
  $\forall$ e,e$'$:E•has_$\mathcal{L}$(e)$\wedge$has_$\mathcal{L}$(e$'$)$\wedge$e$\neq$e$'$ $\Rightarrow$ obs_$\mathcal{L}$(e)$\sqcap$obs_$\mathcal{L}$(e$'$)=$\emptyset$

The location point space, $\mathcal{L}$, the empty location, $\emptyset$, and the operations $=, \neq, \sqcup, \sqcap, \sqsubset$ and $\sqsubseteq$ are defined using standard mathematical topology.

The axiom expresses that two observable, i.e., phenomenological (simple) entities that both enjoy the has_$\mathcal{L}$ property, have distinct, non-overlapping locations.

We consider $\mathcal{L}$ to be an attribute of those phenomena (and concepts) which satisfy the has_$\mathcal{L}$ property.[14]

[12] Yes, in view of the previous statement this is, of course, a "cop out".
[13] The "cop out" is being further compounded!.
[14] For concepts one really should speak of conceptual locations, say ℓ:L, rather than "real" locations.

Universals, that is, attributes of entities (that may enjoy the $\mathsf{has\_}\mathcal{L}$ property), do not satisfy the $\mathsf{has\_}\mathcal{L}$ property.

Examples of observable phenomena, that do not enjoy the $\mathsf{has\_}\mathcal{L}$ property, are: voltage, a person's age, the colour red, etc. □

Atomic and Composite Entities: From now on we restrict our inquiry to discrete, existing independent simple entities.

Atomic Entities: A simple entity is said to be atomic if it cannot be meaningfully decomposed into particulars where these particulars have a useful "value" in the context in which the simple entity is viewed and while still remaining an instantiation of that entity.

The only thing characterising an atomic entity are, to us, its attributes.

Examples: Possible attributes of link simple entities, of the example of Sect.2, could be (i) length; (ii) whether of modality 'road', 'rail track', 'sea lane', 'air lane', or other; (iii) if of modality 'road', then whether a 'dirt road', 'bituminous', 'asphalt', 'concrete', or other; (iv) state-of-repair; (v) exact curve properties; (vi) freeway or toll road; (vii) public or private; (viii) on solid ground, through a tunnel or over a bridge; etcetera. □

Each of these and other kinds of simple entity attributes (i.e., universals) are of conceptual nature, they have no separate existence, but they can be measured (sensed, etc.), and one cannot "remove" a single one of them without the simple entity "loosing" its status as it has been described. We do not rule out that there might be simple entities, of type E, which may possess a proper subset of the attributes of simple entities of another type, E'.

Composite Entities: A simple entity, c, is said to be composite if it can be meaningfully decomposed into sub-entities that have separate meaning, separate existence, in the context in which c is viewed.

We view a composite entity as fully characterised by (i) its attributes, (ii) its sub-entities, and (iii) their mereology (See Sect. 6 on 'Mereology'). That is: We view sub-entities (say the links and hubs of the example in Sect. 2) independent of how they are put together and independent of the attributes of the simple entity of which they are a part (that is, of the net of the example in Sect. 2). And we view a composite simple entity's attributes independent of its sub-entities and independent of the mereology of these sub-entities.

Examples of attributes of discrete, composite simple entities are: *(a) Transport Nets:* (a1) owner of net (name etc. of 'regional', 'state', 'provincial', 'city', etc. 'authority'), (a2) net modality ('freeway', 'toll road', 'canal', 'river', ...), (a3) geographic area of the net (within continent, region, country, province, ...), etcetera. *(b) Container Line:* (b1) administrative information (owner name, addresses, etc.), (b2) line modalities (one or more of 'coastal', 'internal waterway', ocean', or other kinds of sea lanes), (b3) possible container constraints or specialties (one or more of '20 feet', '40 feet', 'non-explosive', 'explosive', 'flammable', 'inflammable', 'reefer', etc.), etcetera. *(c) Pipeline System:* (c1) owner (administrative information: name, addresses, etc.), (c2) gas and/or oil pipeline(s), (c3) geographic area of the net (within continent, region, country, province, ...), etcetera. □

Examples of sub-entities of discrete, composite simple entities are: *(d) Railway Systems:* (d1) net railway (itself a composite transport net, including signals, etc.), (d2) trains (themselves composite: locos, passenger cars, freight cars, etc.), (d3) timetables, (d4) rostering tables, etcetera. *(e) Financial Service Industry:* (e1) financial instruments (coins, notes, stocks, bonds, deeds, etc.), (e2) banks, brokers, traders, commodities exchanges, portfolio managers, insurance companies, etc., (e3) accounts (demand/deposit, mortgage, etc.), (e4) regulatory bodies (in the US: Commodities Futures Trading Commission (CFTC), Federal Deposit Insurance Corporation (FDIC), Federal Reserve Board, Office of the Comptroller of the Currency (OCC), Security & Exchange Commission (SEC)), etcetera. *(f) Logistics:* (f1) freight, (f2) senders and receivers, (f3) logistics firms, (f4) transport companies (trucking, freight train operators, cargo airlines, line, tramp and bulk shipping companies), (f5) transport timetables, (f6) bill-of-lading, way-bill, etc., etcetera. □

5.3.5 Operations, Events and Behaviours

Many of the issues discussed here in connection with simple entities carry over to operations, events and behaviours. We shall however refrain from including such a special study into this paper. Instead we refer to [15, Eir & Bjørner] where we also examine composite operations, events and behaviours.

5.3.6 Discussion

Almost all the statements of Sect. 5.3.4 are not expressed in a *discursive way of reasoning* such as one would express it in a philosophical treatise. Perhaps they ought to be so expressed. For each of these many statements we ought — had this been a philosophical treatise supposedly complementing or in contrast to Russell's thoughtful and reasoned arguments [61–64] — to have discussed alternative ways of looking at simple entities, of their modelling, of relations between narrative English and mathematical formalisations. We, regretfully, dispense with this philosophising, in this paper. Instead you may read Sect. 5.3.4 as the way in which a computing scientist, on the background of 50 years of computer science, interprets Russell's ideas — albeit in a "slightly" (?) narrower context !

6 Mereology

6.1 General

"Mereology (from the Greek $\mu\rho\epsilon o$), 'part') is the theory of part-hood relations: of the relations of part to whole and the relations of part to part within a whole. It is not until Leśniewski's Foundations of a General Theory of Manifolds (1916, in Polish) that a pure theory of part-relations was given an exact formulation. Because Leśniewski's work [55,66] was largely inaccessible to non-readers of Polish, it is only with the publication of Leonard and Goodman's The Calculus of Individuals ([53, Goodman 1940]) that mereology has become a chapter of central interest for modern ontologists and metaphysicians."[15]

Our interest in mereology was first "awoken" by presentations, in the IFIP Working Group 2.3 in the 1980s and 1990s, by the late Douglas Taylor Ross[16].

[15] Stanford Encyclopedia of Philosophy / Wikipedia
[16] http://en.wikipedia.org/wiki/Douglas_T._Ross

We shall now present an example which is claimed to capture an essence of the kind of mereologies that are covered in [22].

6.2 A Model of Mereology

The example is claimed to be generic. When shown as diagrams, the boxes–within–boxes and the "fat", black connectors that "criss-cross" boxes — of Fig. 1 — can be shown to "mimic" the structure of such infrastructure components.

Examples: air traffic, a financial service industry, a pipeline system, a railway system, etcetera. Similarly the formula parts (for boxes and connectors) thus relate to phenomena of such systems. For air traffic some boxes are aircraft, others are ground or terminal control towers, yet others are regional and continental control centers, etc. Connectors of air traffic are the radio-telephone paths that allow communication between air traffic equipment and staff. For a railway system some boxes are train stations, others are railway tracks (lines) between stations — with stations and lines consisting of embedded boxes in the form of rail units: linear, switches, crossovers, etc. Connectors compose boxes into meaningful track layouts and signalling paths. □

We speak of systems, s:S, as *assemblies* (below referred to as a:A). From an assembly we can immediately observe, obs_Ps, a set of particulars. Particulars are either assemblies or *units*. For the time being we do not further define what units are.

type
  S = A, U, P = A | U
value
  obs_Ps: A → P-**set**

Particulars observed from a assembly are said to be immediately *embedded* (or *within*) in that assembly; and two or more particulars observed from an assembly are said to be immediately *adjacent* to one another.

Figure 1 illustrates a hypothetical composite entity[17].

Embeddedness and adjacency generalises to transitive relations.

All particulars observable from a system are distinct.

Given obs_Ps we can define a function, xtr_Ps, which applies to an assembly, a, and which extracts all particulars properly embedded in a. The functions obs_Ps and xtr_Ps define the meaning of *proper embeddedness*.

value
  xtr_Ps: A → P-**set**
  xtr_Ps(a) ≡
    let ps = obs_Ps(a) **in** ps ∪ ∪{xtr_Ps(a′)|a′:A•a′ ∈ ps} **end**

Particulars have *unique identifiers*, PI.

[17] Figures 1 on the next page and 2 on page 18 only serve to illustrate the formulas. In a domain description we do not give instances of figures of simple entities such as the generic ones here. Instead we present sorts (as here, where L, H, LI, HI, PI, K and KI are sorts, and types (as here, where P is defined in terms of L and H, and where axioms using observer functions impose an appropriate syntax on how simple entities are composed.

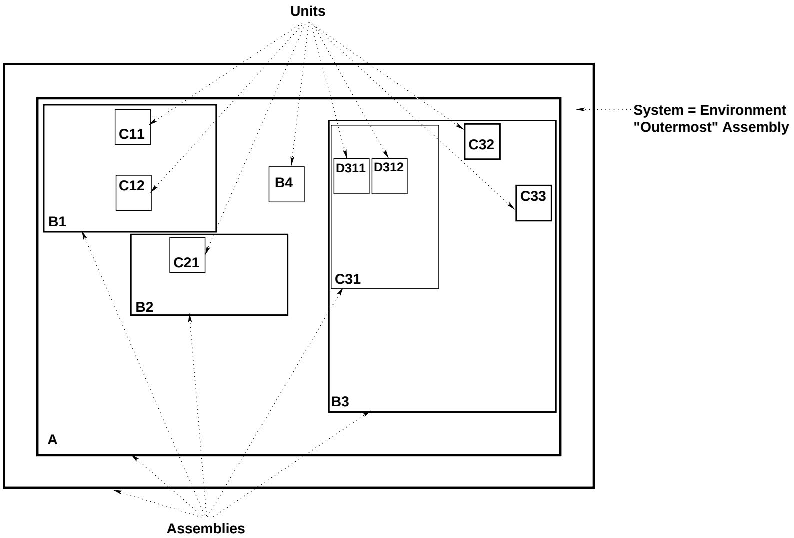

Fig. 1 Assemblies and units embedded in an *environment*

type
  PI
value
  obs_PI: P → PI
axiom
  ∀ a:A •
    let ps = obs_Ps(a) **in**
    ∀ p′,p″:P • {p′,p″}⊆ps ∧ p′≠p″ ⇒ obs_PI(p′)≠obs_PI(p″) ∧
    ∀ a′,a″:A • {a′,a″}⊆ps ∧ a′≠a″ ⇒ xtr_Ps(a′)∩ xtr_Ps(a″)={} **end**

We shall now add to this a rather general notion of particulars being otherwise related. That notion is one of *connectors*, k:K.

Connectors may, and usually do provide for connections — between particulars. A connector is an ability be be connected. A connection is the actual fulfillment of that ability. Connections are relations between two particulars. Connections "cut across" the "classical" *particulars being part of the (or a) whole* and *particulars being related by embeddedness or adjacency*.

Figure 2 "repeats" Fig. 1 but "adds" connectors. The idea is that connectors allow a assembly to be connected to any embedded particular, and allow any two (transitively) adjacent particulars to be connected.

In Fig. 2 assembly A is connected, by *K2*, (without, as we shall later see, interfering with assembly B1), to part C11; the "outermost" assembly is connected, by K1 to B1, etcetera.

From a system, i.e., an assembly, we can observe, obs_Ks, all its connectors. From a connector we can observe, obs_KI, its unique connector identifier, KI, and the set of two (like an unordered pair of) identifiers, obs_PIp, of the particulars that the connector connects, All particular identifiers of system connectors identify particulars of the system. All observable connector identifiers of particulars identify connectors of the system.

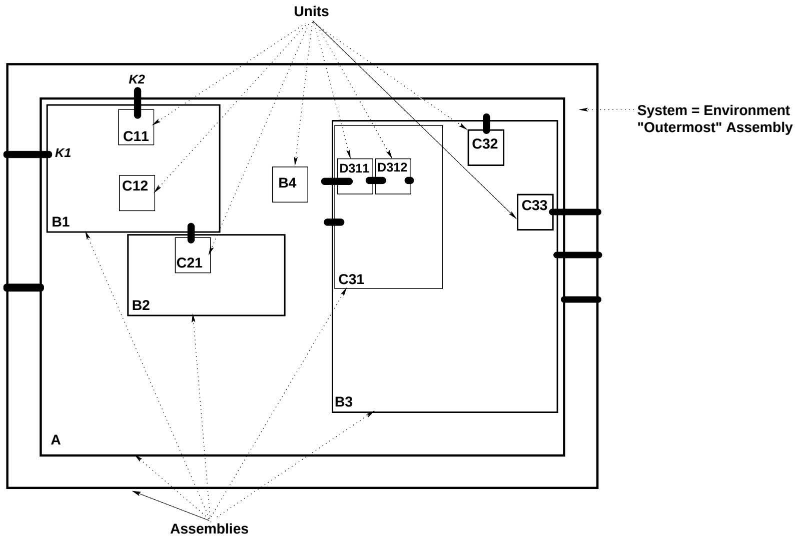

Fig. 2 Assembly and unit connectors

type
  K
value
  obs_Ks: S → K-**set**
  obs_KI: K → KI
  obs_PIp: K → PI-**set**
  obs_KIs: P → KI-**set**
axiom
  $\forall$ s:S,k:K • k $\in$ obs_Ks(s) $\Rightarrow$ $\exists$ p:P • p $\in$ xtr_Ps(s) $\Rightarrow$ obs_PI(p) $\in$ obs_PIp(k),
  $\forall$ s:S,p:P • $\forall$ ki:KI • ki $\in$ obs_KIs(p) $\Rightarrow$ $\exists!$ k:K • k $\in$ obs_Ks(s) $\wedge$ ki=obs_KI(k)

This model allows for a rather "free-wheeling" notion of connectors one that allows internal connectors to "cut across" (even transitively) embedded and (transitively) adjacent particulars.

We may need to define an auxiliary function. xtr∀KIs(p) applies to a system and yields all its connector identifiers.

value
  xtr∀KIs: S → KI-**set**
  xtr∀Ks(s) ≡ {obs_KI(k)|k:K•k $\in$ obs_Ks(s)}

There is an aspect of assemblies and units that we have not mentioned. You may, "mereologically speaking", think of that aspect as the "white space" within the borders of a unit box, and within the borders of an assembly box exclusive of that assembly box's embedded assemblies and units. Let us associate a notion of state, i.e., a state, that is, set of static and dynamic attributes of assemblies and units, with that "white space" !

type
  Σ

value
obs_Σ: P $\rightarrow$ Σ

In axiomatic mereology there is a notion of overlap. Overlap, to us, has to do with two "white spaces" of distinct particulars. Connectors between such particulars can then be interpreted as also providing for overlaps.

The above model is now claimed to be a model of axiom systems proposed in the literature [22].

• • •

This ends our model of a concept of mereology. The particulars are those of assemblies and units. The relations between particulars and the whole are, on one hand, those of embeddedness and adjacency, and on the other hand, those expressed by connectors: relations between arbitrary `part`iculars.

6.3 Relation to The Casati Varzi Mereology [22]

6.3.1 "Classical" Mereology Operators

Typically `part/whole` relations for mereologies in [22] are:

- $\mathcal{P}xy$: x is a `part` of y,
- $\mathcal{PP}xy$: x is a proper `part` of y,
- $\mathcal{O}xy$: x and y overlaps and
- $\mathcal{U}xy$: x and y underlap.

We refer to [22] for axioms over these operators. We now give but one of many possible interpretations of the above `part/whole` relations. The interpretation is with respect to the model given in Sect. 6.2. We apologize for the "double" use of the term 'part': when we use the *slanted sans serif font 'part'* we refer to a p:P, and when we use the `teletype font 'part'` we me the relationship designated by $\mathcal{P}$ or $\mathcal{PP}$. The mereology operator $\mathcal{P}$ is taken as a primitive. The other mereology operators, $\mathcal{PP}$, $\mathcal{O}$ and $\mathcal{U}$ are defined in terms of $\mathcal{P}$.

- $\mathcal{P}xy$: x is a `part` of y.[18]
  - $\star$ A *part* is a `part` of itself; attributes of a *part* are `part`s of that *part*.[19]
- $\mathcal{PP}xy$: x is a proper `part` of y: $\mathcal{PP}xy \leftrightarrow (\mathcal{P}xy \wedge \sim \mathcal{P}xy)$.
  - $\star$ *Part*s of an assembly are proper `part`s of that assembly. Any proper subset of attributes of a *part* are proper `part`s of that *part*.
- $\mathcal{O}xy$: x and y overlaps: $\mathcal{O}xy \leftrightarrow \exists z[\mathcal{P}zx \wedge \mathcal{P}zy]$.
  - $\star$ Two *part*s that are connected overlap. If two or more attributes of a *part* are not (mutually) independent, that is, these dependent attribute values stand in some relation to one another, then these attributes overlap.
- $\mathcal{U}xy$: x and y underlap if there exists an object z such that x and y are both `part`s of z: $\mathcal{U}xy \leftrightarrow \exists z[\mathcal{P}xz \wedge \mathcal{P}yz]$
  - $\star$ If two or more *part*s are contained in some assembly, a, then these contained *part*s underlap with a.

[18] The • (bulleted) statement(s) expresses an informal definition of the mereology operator.
[19] The $\star$ (starred) statement(s) interprets the • (bulleted) statement(s) in terms of the model of mereology given in Sect. 6.2.

6.3.2 Discussion of Our Interpretation

First we remind the reader that we have given but one of many possible interpretations of the `part/whole` relations.

We could have given other interpretations; for example: (i) we could have omitted any reference to *part* attributes; (ii) we could have restricted `overlap` to relate to only specifically characterised *connectors*; (iii) or both (i–ii).

The scope of our discussion will now be enlarged to cover other kinds of mereologies.

6.4 Discussion

We have partially covered the mereological system of [22, Casati & Varzi, 1999]. There are other mereological systems.

In [23, Bowman L. Clarke, 1981] (*A Calculus of Individuals Based on 'Connection'*) a mereology operator, $\mathcal{C}$, for 'connected'[20], is the primitive in terms of which the above operators ($\mathcal{P}, \mathcal{PP}, \mathcal{O}$, and $\mathcal{U}$) can be defined. The intuition behind the $\mathcal{C}$onnected operator, however, makes it of less interest to us: that intuition implies a notion of points which we do not wish to bring into our models of man-made infrastructure components. Bowman L. Clarke's system allows definition of external and internal, as well as tangential connectedness. [24, Bowman L. Clarke, 1985] (*Individuals and Points*) further elaborates on this, for our purpose, "uninteresting" concept of points.

In [72, Achille C. Varzi, 1993] (*Spatial Reasoning in a Holey*[21] *World*), in [71, Achille C. Varzi, 1994], (*On the Boundary between Mereology and Topology*) and in [65, Barry Smith, 1996] (*Mereo-topology: A Theory of Parts and Boundaries*), extensions to the operators covered above (i.e., ($\mathcal{P}, \mathcal{PP}, \mathcal{O}$, and $\mathcal{U}$) are applied to cover topological spaces. It would be interesting to study these papers further in order to determine their relevance to the modelling of the man-made infrastructure components in which we are interested.

In closing this section on 'mereology' we claim (i) that our model of mereology is in line with main-stream axiom systems for the kind of mereologies that Stanisław Leśniewski studied, (ii) that that model covers of a large class infrastructure components which we have so far modelled[22], and (iii) that, apparently, we do not need a more sophisticated concept of mereology than covered by [22].

7 Bertrand Russell's Philosophy of Logical Atomism

7.1 A Metaphysics and a Methodology

Russell described his philosophy, which he referred to as 'Logical Atomism', as based on both a *metaphysical* view and a way (a *methodology*) of doing philosophy.

We refer to the indented 'metaphysics' and the 'methodology' paragraphs of Sect. 1.1.2 (Pages 3–3) for their characterisation.

---

[20] The $\mathcal{C}$onnected operator is not to be confused with the connectors, k:K, of our model.

[21] holey: something full of holes

[22] [4, The Market], [19, Railways], [9, Financial Service Industry], [8, A Container Line Industry], [12, Logistics], [11, Oil Pipelines], etcetera.

The reason we are interested in Russell's 'Logical Atomism' is that its *metaphysical* view and its *methodology* — although meant for philosophical inquiry — very much resembles our view of the metaphysics of domains and the method by which domain analysts analyse.

We refer to [64, Vol. 8, Part III, Chap. 17, pp 157–244, same as [63]]. Russell's writings on 'Logical Atomism' contains several sub-topics. We shall next examine some of these.

7.2 A Logically Ideal Language

The analysis part of the methodology, Russell claimed, would eventually result in a minimal language containing only words for simple particulars and concepts, their simple properties, relations amongst these and logical constants — a language which could adequately capture all truths; that is, other particulars and concepts can be defined and the most general and basic principles can be derived. In the minimal language the simplest of complete sentences, containing just a single predicate or verb representing a quality (i.e., an attribute) or a relation over simple entities, would be what Russell called *atomic propositions*. The truth of an atomic proposition then depends on a single atomic fact. *Molecular propositions* are then formed by combining atomic propositions using the logical connectives. *Existential* or *general propositions* are formed by replacing proper constituents of simpler propositions by variables and prefixing a universal or an existential quantifiers. Hence atomic facts are at the core of Russell's metaphysics.

Instead of first using only informal language to narratively describe domain facts we advocate, motivated by Russell's Logical Atomism, both using precise formal language and to couple statements in that formal language to informal narratives — each in their way describing the domain. In our case we use a partly algebraic, that is a logical language over sorts, partly a model-oriented language over mathematical structures such as sets, Cartesians, lists, maps (i.e., finite definition set functions) and functions.

7.3 A Monadistic View of Domain Modelling

Russell opposed the "doctrine of internal relations" (every relation is grounded in the natures of the related terms) and instead expressed [62, Page 221] *"it is a common opinion ... that all propositions ultimately consist of subject and predicate"*. An interpretation of this is: *"Given, say, the proposition aRb, where R is some relation, this monadistic view will analyse this into two propositions, which we may call* a_{r_1} *and* b_{r_2}*, which give to a and b respectively, adjectives supposed to be together equivalent to R."* This monadistic view is analysed in [26, J.G. Nilsson: On Reducing Relationships to Property Ascriptions] where it is *"argued that relationships may preferably be represented formally as property ascriptions."*

In Sect. 5.3.4 the monadistic view was adopted: With atomic and composite simple entities, a and b, we associated attributes, including the mereological ones of composite attributes. Let two such simple entities, a and b, be connected by a "possessing" an identifier, i_b of b, and b "possessing" an identifier, i_a of a, as in the example of Sect. 2. The subjects are a and b, and the constants of a_{r_1} and b_{r_2} are i_b, respectively i_a. Thus the mereological view supports the monadistic view when expressing properties of composite entities. (Russell uses the term 'complex' where we use the term 'composite'.)

7.4 Discussion

There are other facets to Russell's work on Logical Atomism. It seems, however, to this author, that the issues we have covered in this section and which have otherwise been covered in previous and the next section suffice for now.

Our preliminary conclusions are twofold: (i) whereas mereology has very clear and focused "uses" in domain science, (ii) the "uses" of Logical Atomism is of a more meta-conceptual nature, that is, as brought out in Sect. 1.1.2's two (indented) paragraphs on metaphysics and methodology.

8 On Laws of Domain Descriptions

8.1 Preliminaries

8.1.1 Suppression of Unique Identification

When comparing, for example, two simple entities one is comparing not only their attributes but also, when the entities are composite, their sub-entities. Concerning unique identifiers of simple entities we have this to say: We can decide to either include unique identifiers as an entity attribute, or we can decide that such identifiers form a third kind of observable property of a simple entity the two others being ("other") attributes — as we see fit to define and the possible sub-entities of composite entities.

Either way, we need to introduce a meta-linguistic operator[23], say

$$\mathcal{S}_I\text{: Simple\_observable\_entity\_value} \rightarrow \text{Anonymous\_simple\_entity\_value}$$

The concept of an anonymous value is also meta-linguistic. The anonymous value is basically "the same, i.e., "identical" value as is the simple entity value (from which, through $\mathcal{S}_I$, it derives)[24] with the single exception that the simple entity value "possesses" the unique identifier of the observable entity value and the anonymous entity value does not.

8.1.2 Distinguishability of Simple Entities

When we wish to distinguish one simple entity phenomenon from another then we say that the two ("the one and the other") are distinct. To be distinct to us means that the two phenomena have distinct, that is, unique identifiers. Being simple entity phenomena, separately observable in the domain, means that their spatial (positional) properties are distinct. That is their anonymous values are distinct. Meta-linguistically, that is, going outside the `RSL` framework[25], we can "formalise" this:

[23] The operator $\mathcal{S}_I$ is meta-linguistic with respect to `RSL`: it is not part of `RSL`, but applies to `RSL` values.

[24] The $\mathcal{S}$ stands for "suppress" and the $\mathcal{I}$ for the suppressed unique identifier.

[25] but staying within a proper mathematical framework — once we have understood the mathematical properties of $\mathcal{S}_I$ and proper `RSL` values and 'anonymous' values (which, by the way, are also `RSL` values)

type
  E [A models a type of simple entity phenomena]
  I^{26} [I models the type of unique E identifiers]
value
  obs_I: E → I
axiom
  $\forall$ e,e′:E • obs_I(e)≠obs_I(e′) ⇒ $\mathcal{S}_I$(e)≠$\mathcal{S}_I$(e′)

The above applies to any kind of observable simple entity *phenomenon* A. It does not necessarily apply to observable simple entity *concepts*. **Example:** *Two uniquely identified timetables may have their anonymous values be the exact same value.*

Simple entity phenomena, in our ontology, are closely tied to space — and perhaps even space/time "co-ordinates" — with no two simple entity phenomena sharing overlapping space. Concepts are, in our ontology, not so constrained, that is, we allow "copies" although uniquely named !

8.2 Some Laws

Sections 5–7 and the material of this section, Sect. 8.1 above, can be summarised by proposing a number of domain description laws. We shall just bring a few of these laws here. Enough, we hope, to spur further research into 'laws of description'.

8.2.1 Unique Identifiers

Domain Description Law 1, Unique Identifiers: *If two observable simple entities have the same unique identifier then they are the same simple entity.* □

Any domain description must satisfy this law. The domain describer must, typically through axioms, secure that the domain description satisfy this law. Thus there is a *proof obligation* to be *dispensed*, namely that the unique identifier law holds of a domain description.

8.2.2 Unique Phenomena

Domain Description Law 2, Unique Phenomena: *If two observable simple entities have different unique identifiers then their values, "stripped" of their unique identifiers are different.* □

Any domain description must satisfy this law. The domain describer must, typically through axioms, secure that the domain description satisfy this law. Thus there is a *proof obligation* to be *dispensed*, namely that the unique phenomena law holds of a domain description.

[26] We have here emphasized I, the type name of the type of unique A identifiers. Elsewhere in this book we treat types of unique identifiers of different types of observable simple entities as "ordinary" RSL types. Perhaps we should have "singled" such unique identifier type names out with a special font ? Well, we'll leave it as is !

8.2.3 Space Phenomena Consistency

Domain Description Law 3, Space Phenomena Consistency: *Two otherwise unique, and hence distinctly observable phenomena can, spatially, not overlap.* □

We can express the *Space/Time Phenomena Consistency Law* meta-linguistically, yet in a proper mathematical manner:

type
  E [E is the type name of a class of observable simple entity phenomena]
  I [I is the type name of unique E identifiers]
  $\mathcal{L}$ [$\mathcal{L}$ is the type name of E locations]
value
  obs_I: E → I
  obs_$\mathcal{L}$: E → $\mathcal{L}$
axiom
  ∀ e,e′:E • obs_I(e)≠obs_I(e′) ⇒ obs_$\mathcal{L}$(e) ⊓ obs_$\mathcal{L}$(e′) = ∅

We can assume that this law always holds for otherwise unique, and hence distinctly observable phenomena.

8.2.4 Space/Time Phenomena Consistency

Domain Description Law 4, Space/Time Phenomena Consistency: Monotonicity: *If an entity (that has the location property), at time t is at location ℓ, and at time t' (larger than t) is at location ℓ' (different from ℓ), then it moves monotonically from ℓ to ℓ' during the interval (t, t').* □

Specialisations of this law are, for example, that if the movement is of two simple entities, like two trains, along a single rail track and in the same direction, then where train s_i is in front of train s_j at time t, train s_j cannot be in front of train s_i at time t' (where $t' - t$ is some small time interval).

8.3 Discussion

There are more laws. And there are most likely laws that have yet to be “discovered” ! Any set of laws must be proven consistent. And any domain description must be proven to adhere to these (and “the” other) laws.

We decided to bring this selection of laws because they are a part of the emerging ‘domain science’.

Laws 3 and 4 are also mentioned, in some other form, in [63].

9 Conclusion

We have examined some facets of domain science and related them to mereology and logical atomism. The surface has been just scraped. There is much more work to do. Both this paper and that of [15, Bjørner & Eir] thus broaches topics that are still very much also in the realm of philosophy — as the many references of this paper to philosophical treatises bears witness to. In fact it may very well be time to recognise the need for much more research into a, or the philosophy of computer science. Topics of study within such a philosophy of computer science could include: *what are the sources of computer science subject matter, is Church's Thesis adequate for the understanding of computing,* cf. [25], *what is the ontological status of domain entities, what is the rôle of hermeneutics in computer science, what kinds of inquiry play a rôle in computer science, what are the objectives of computer science inquiry, what gives computer science its hold on experience, what are the human traits behind computer science, what is computer science beauty, what is the relationship between the abstract world of computer science and the material universe* and *how do we know whether a computer science proof is correct?* etcetera. Much, but not all, of this borders strongly to philosophy of mathematics.

Acknowledgements My awareness of Stanisław Leśniewski's work on mereology was due to (the late) Doug Ross' talks at IFIP Working Group 2.3 meetings in the 1980s and 1990s – spurred on by Mike Woodger's interactions with Doug Ross. I studied Bertrand Russell's work on Theory of Descriptions years ago, but now wish to thank Tony Hoare for his steering me in the direction of Russell's Philosophy of Logical Atomism.

Homage The author's relation to Peter Landin is sôlely through Landin's publications [42, 43, 47–49, 51, 50] and his many internal reports, notably from the Univac Systems Programming Group (in New York, USA) [44–46, a few are cited here]. On Pages 88–90 in [6, Sect. 3.5.2] (a monograph cum textbook) I briefly overview some of Peter Landin's pioneering contributions to semantics (of formal languages). Landin's work deeply influenced the work of the IBM Vienna Laboratory from the early 1960s to the early/mid 1970s, viz. `VDM` [17, 18, 28, 27]. Over more than 30 years of teaching MSc and PhD courses my students have all been exposed to Landin's SECD machine, to his elegant explanation of (λ-Calculus, i.e., syntactic) fix-points, to his continuation operator J ([45, 70]) — as well as to some of Landin's other thoughts (for example [51]). Many of my former students also know well the meaning of the acronym `ISWIM`! Their understanding of *"first semantics, then syntax"* is well-cemented.

References

1. Abrial, J.-R.: 1996, *The B Book: Assigning Programs to Meanings*, Tracts in Theoretical Computer Science. Cambridge, England: Cambridge University Press.
2. Abrial, J.-R.: 2009, *Modeling in Event-B: System and Software Engineering.* Cambridge, England: Cambridge University Press.
3. Aristotle: 1933 (1996), *Metaphysics, Books I-IX*, Loeb Classical Library No. 271. Cambridge, Mass., USA: Harvard University Press.
4. Bjørner, D.: 2002, 'Domain Models of "The Market" — in Preparation for E–Transaction Systems'. In: *Practical Foundations of Business and System Specifications (Eds.: Haim Kilov and Ken Baclawski).* The Netherlands.
5. Bjørner, D.: 2006a, *Software Engineering, Vol. 1: Abstraction and Modelling*, Texts in Theoretical Computer Science, the EATCS Series. Springer.
6. Bjørner, D.: 2006b, *Software Engineering, Vol. 2: Specification of Systems and Languages*, Texts in Theoretical Computer Science, the EATCS Series. Springer. Chapters 12–14 are primarily authored by Christian Krog Madsen.
7. Bjørner, D.: 2006c, *Software Engineering, Vol. 3: Domains, Requirements and Software Design*, Texts in Theoretical Computer Science, the EATCS Series. Springer.
8. Bjørner, D.: 2007a, 'A Container Line Industry Domain'. Techn. report, http://www2.imm.dtu.dk/~db/container-paper.pdf, Fredsvej 11, DK-2840 Holte, Denmark.
9. Bjørner, D.: 2007b, 'A Financial Services Industry Domain'. Techn. report, http://www2.imm.dtu.dk/~db/fsi.pdf, Fredsvej 11, DK-2840 Holte, Denmark.

10. Bjørner, D.: 2008, ‘From Domains to Requirements’. In: *Montanari Festschrift*, Vol. 5065 of *Lecture Notes in Computer Science (eds. Pierpaolo Degano, Rocco De Nicola and José Meseguer)*. Heidelberg, pp. 1–30.
11. Bjørner, D.: 2009a, ‘A Domain Model of Oil Pipelines’. Techn. report, http://www2.imm.dtu.dk/˜db/pipeline.pdf, Fredsvej 11, DK-2840 Holte, Denmark.
12. Bjørner, D.: 2009d, ‘What is Logistics ? A Domain Analysis’. Techn. report, http://www2.imm.dtu.dk/˜db/pipeline.pdf, Fredsvej 11, DK-2840 Holte, Denmark.
13. Bjørner, D.: March 2009b, *Domain Engineering: Technology Management, Research and Engineering*. JAIST Press. JAIST Research Monograph # 4, 536 pages.
14. Bjørner, D.: Summer 2009c, *From Domains to Requirements: The Triptych Approach to Software Engineering*. Submitted to Springer for evaluation in 2009. Slightly incomplete draft version, approximately XXVII+160+25 pages (frontmatter, main text, appendices). See http://www.imm.dtu/˜db/de+re-p.pdf.
15. Bjørner, D. and A. Eir: 2008, *Compositionality: Ontology and Mereology of Domains. Some Clarifying Observations in the Context of Software Engineering in Festschrift for Prof. Willem Paul de Roever, eds. Martin Steffen, Dennis Dams and Ulrich Hannemann*, Lecture Notes in Computer Science. Heidelberg: Springer.
16. Bjørner, D. and M. C. Henson (eds.): 2008, *Logics of Specification Languages*, EATCS Series, Monograph in Theoretical Computer Science. Heidelberg, Germany: Springer.
17. Bjørner, D. and C. B. Jones (eds.): 1978, *The Vienna Development Method: The Meta-Language*, Vol. 61 of *LNCS*. Springer–Verlag.
18. Bjørner, D. and C. B. Jones (eds.): 1982, *Formal Specification and Software Development*. Prentice-Hall.
19. Bjørner, D. and M. Pěnička: 2004, ‘Towards a TRAIN Book for The RAIlway DomaiN’. Techn. reports, http://www.railwaydomain.org/PDF/tb.pdf, The TRAIN Consortium.
20. Cansell, D. and D. Méry: 2008, *Logics of Specification Languages*, Chapt. The event-B Modelling Method: Concepts and Case Studies, pp. 47–152 in [16]. Springer.
21. Carnap, R.: 1928, *Der Logische Aufbau der Welt*. Berlin: Weltkreis.
22. Casati, R. and A. Varzi: 1999, *Parts and Places: the structures of spatial representation*. MIT Press.
23. Clarke, B. L.: 1981, ‘A Calculus of Individuals Based on ‘Connection’’. *Notre Dame J. Formal Logic* **22**(3), 204–218.
24. Clarke, B. L.: 1985, ‘Individuals and Points’. *Notre Dame J. Formal Logic* **26**(1), 61–75.
25. Dershowitz, N. and Y. Gurevich: 2008, ‘A Natural Axiomatization of Computability and Proof of Church’s Thesis’. *The Bulletin of Symbolic Logic* **14**(3), 299–350.
26. Fischer Nilsson, J., *On Reducing Relationships to Property Ascriptions*, Vol. 190 of *Frontiers in Artificial Intelligence and Applications*, Chapt. Information Modelling and Knowledge Bases # XX. IOS Press, Amsterdam, The Netherlands, eds. Y. Kiyoki, T. Tokuda, H. Jaakkola, X. Chen and N. Yoshida.
27. Fitzgerald, J. S.: 2008, *Logics of Specification Languages*, Chapt. The Typed Logic of Partial Functions and the Vienna Development Method, pp. 453–487 in [16]. Springer.
28. Fitzgerald, J. S. and P. G. Larsen: 1997, *Developing Software using VDM-SL*. The Edinburgh Building, Cambridge CB2 1RU, England: Cambridge University Press.
29. George, C. and A. E. Haxthausen: 2008, *Logics of Specification Languages*, Chapt. The Logic of the RAISE Specification Language, pp. 349–399 in [16]. Springer.
30. George, C. W., A. E. Haxthausen, S. Hughes, R. Milne, S. Prehn, and J. S. Pedersen: 1995, *The RAISE Method*, The BCS Practitioner Series. Hemel Hampstead, England: Prentice-Hall.
31. Goodman, N., *The Structure of Appearance*. 1st.: Cambridge, Mass., Harvard University Press; 2nd.: Indianapolis, Bobbs-Merrill, 1966; 3rd.: Dordrecht, Reidel, 1977.
32. Goodman, N.: 1990, *A Study of Qualities*. New York: Garland. Ph.D. dissertation thesis, Harvard, 1940.
33. Harel, D.: 1987, ‘Statecharts: A Visual Formalism for Complex Systems’. *Science of Computer Programming* **8**(3), 231–274.
34. Henson, M. C., M. Deutsch, and S. Reeves: 2008, *Logics of Specification Languages*, Chapt. Z Logic and Its Applications, pp. 489–596 in [16]. Springer.
35. ITU-T: 1992, ‘CCITT Recommendation Z.120: Message Sequence Chart (MSC)’.
36. ITU-T: 1996, ‘ITU-T Recommendation Z.120: Message Sequence Chart (MSC)’.
37. ITU-T: 1999, ‘ITU-T Recommendation Z.120: Message Sequence Chart (MSC)’.
38. Jackson, D.: April 2006, *Software Abstractions Logic, Language, and Analysis*. Cambridge, Mass., USA: The MIT Press. ISBN 0-262-10114-9.

39. Jackson, M. A.: 1995, *Software Requirements & Specifications: a lexicon of practice, principles and prejudices*, ACM Press. Wokingham, nr. Reading, England; E-mail: ipc@awpub.add-wes.co.uk: Addison-Wesley Publishing Company. ISBN 0-201-87712-0; xiv + 228 pages.
40. Jackson, M. A.: 2001, *Problem Frames — Analyzing and Structuring Software Development Problems*, ACM Press, Pearson Education. Edinburgh Gate, Harlow CM20 2JE, England: Addison–Wesley.
41. Lamport, L.: 2002, *Specifying Systems*. Boston, Mass., USA: Addison–Wesley.
42. Landin, P. J.: 1964, 'The Mechanical Evaluation of Expressions'. *Computer Journal* **6**(4), 308–320.
43. Landin, P. J.: 1965a, 'A Correspondence Between ALGOL 60 and Church's Lambda-Notation (in 2 parts)'. *Communications of the ACM* **8**(2-3), 89–101 and 158–165.
44. Landin, P. J.: 1965b, 'A Generalization of Jumps and Labels'. Technical report, Univac Sys. Prgr. Res. Grp., N.Y.
45. Landin, P. J.: 1965c, 'A Generalization of Jumps and Labels'. Research report, UNIVAC Systems Programming Research. Reprinted in Higher-Order and Symbolic Computation 11(2):125–143, 1998, with a foreword [70].
46. Landin, P. J.: 1965d, 'Getting Rid of Labels'. Technical report, Univac Sys. Prgr. Res. Grp., N.Y.
47. Landin, P. J.: 1966a, 'A Formal Description of ALGOL 60'. In: *[67]*. pp. 266–294.
48. Landin, P. J.: 1966b, 'A Lambda Calculus Approach'. In: L. Fox (ed.): *Advances in Programming and Non-Numeric Computations*. Pergamon Press, pp. 97–141.
49. Landin, P. J.: 1966c, 'The Next 700 Programming Languages'. *Communications of the ACM* **9**(3), 157–166.
50. Landin, P. J.: 1998, 'A Generalization of Jumps and Labels'. *Higher-Order and Symbolic Computation* **11**(2), 125–143. Reprinted from a technical report, UNIVAC Systems Programming Research (1965), with a foreword [70].
51. Landin, P. J. and R. Burstall: 1972, 'Programs and their Proofs: An Algebraic Approach'. *Machine Intelligence* **4**.
52. Lejewski, C.: June, 1983, 'A note on Leśniewksi's Axiom System for the Mereological Notion of Ingredient or Element'. *Topoi* **2**(1), 63–71.
53. Leonard, H. S. and N. Goodman: 1940, 'The Calculus of Individuals and its Uses'. *Journal of Symbolic Logic* **5**, 45–44.
54. Leśniewksi, S.: 1927-1931, '0 Podstawack Matematyki (Foundations of Mathematics)'. *Prezeglad Filosoficzny* **30-34**.
55. Luschei, E.: 1962, *The Logical Systems of Leśniewksi*. Amsterdam, The Netherlands: North Holland.
56. Merz, S.: 2008, *Logics of Specification Languages*, Chapt. The Specification Language TLA$^+$, pp. 401–451 in [16]. Springer.
57. Reisig, W.: 1985, *Petri Nets: An Introduction*, Vol. 4 of *EATCS Monographs in Theoretical Computer Science*. Springer Verlag.
58. Reisig, W.: 1992, *A Primer in Petri Net Design*. Springer Verlag. 120 pages.
59. Reisig, W.: 1998, *Elements of Distributed Algorithms: Modelling and Analysis with Petri Nets*. Springer Verlag. xi + 302 pages.
60. Russel, B.: 1952, *"Preface," Our Knowledge of the External World*. London: G. Allen & Unwin, Ltd.
61. Russell, B.: 1905, 'On Denoting'. *Mind* **14**, 479–493.
62. Russell, B.: 1910, *The Principles of Mathematics*. Allen and Unwin, London.
63. Russell, B.: 1918–1919, 'The Philosophy of Logical Atomism'. *The Monist: An International Quarterly Journal of General Philosophical Inquiry,* **xxxviii–xxix**, 495–527, 32–63, 190–222, 345–380.
64. Slater, J. G. (ed.): 1986, *The Collected Papers of Bertrand Russel*. Allen and Unwin, London, England.
65. Smith, B.: 1996, 'Mereotopology: A Theory of Parts and Boundaries'. *Data and Knowledge Engineering* **20**, 287–303.
66. Srzednicki, J. and Z. Stachniak (eds.): 1988, *Leśniewksi's Lecture Notes in Logic*. Dordrecht.
67. Steel, T. B. (ed.): 1966, *Formal Language Description Languages,* IFIP TC-2 Work. Conf., Baden. North-Holland Publ.Co., Amsterdam.
68. Tarski, A., *Appendix E.* [77].

69. Tarski, A.: 1956, *Foundations of the geometry of solids*. Oxford: Oxford University Press. Transl. J. H. Woodger,.
70. Thielecke, H.: 1998, 'An introduction to Landin's "A generalization of jumps and labels"'. *Higher-Order and Symbolic Computation* **11**(2), 117–124.
71. Varzi, A. C.: 1994a, *On the Boundary between Mereology and Topology*, pp. 419–438. Vienna: Hölder-Pichler-Tempsky.
72. Varzi, A. C.: 1994b, *Spatial Reasoning in a Holey*[27] *World*, Vol. 728 of *Lecture Notes in Artificial Intelligence*, pp. 326–336. Springer.
73. Whitehead, A.: 1920, *The Concept of Nature*. Cambridge: Cambridge University Press.
74. Whitehead, A.: 2929, *An Enquiry Concerning the Principles of Natural Knowledge*. Cambridge: Cambridge University Press.
75. Whitehead, A. N. and B. Russell: 1910, 1912, and 1913, *Principia Mathematica, 3 vols.* Cambridge University Press. Second edition, 1925 (Vol. 1), 1927 (Vols 2, 3), also Cambridge University Press, 1962.
76. Woodcock, J. C. P. and J. Davies: 1996, *Using Z: Specification, Proof and Refinement*. Prentice Hall International Series in Computer Science.
77. Woodger, J.: 1937, *The Axiomatic Method in Biology*. Cambridge: Cambridge University Press.
78. Zhou, C. C. and M. R. Hansen: 2004, *Duration Calculus: A Formal Approach to Real–time Systems*, Monographs in Theoretical Computer Science. An EATCS Series. Springer-–Verlag.
79. Zhou, C. C., C. A. R. Hoare, and A. P. Ravn: 1992, 'A Calculus of Durations'. *Information Proc. Letters* **40**(5).

[27] holey: something full of holes

Contents