DOMAIN SCIENCE & ENGINEERING

The TU Wien Lectures, Fall 2022

Dines Bjørner

Technical University of Denmark

The Triptych Dogma

In order to *specify* **software**,
we must understand its requirements.

In order to *prescribe* **requirements**
we must understand the **domain**.

So we must **study, analyse** and **describe** domains.

Lecture 3: Unique Identifiers and Mereology

- We now present a properly systematic treatment of some of the
  - principles,
  - procedures,
  - techniques and
  - tools

  of the *Domain Engineering* method, namely for
  - the **internal qualities** of endurants:
    - unique identification,
    - mereology,
    - attributes and
    - intentional pull.

- Please consider Fig. 4.1 on Slide 63.
  - The previous chapter covered the tree-like structure to the left in Fig. 4.1.
  - This chapter covers the horisontal and vertical lines, also to the left in Fig. 4.1.

• • •

- In this chapter we introduce
  - the concepts of internal qualities of endurants and universal qualities of domains,
  - and cover, first, the analysis and description of internal qualities:
    * unique identifiers (Sect. 5.2 on Slide 178),
    * mereologies (Sect. 5.3 on Slide 202) and
    * attributes (Sect. 5.4 on Slide 230),

– There is, additionally, three universal qualities:
  * space, time (Sect. 5.5 on Slide 293) and
  * intentionality (Sect. 5.6 on Slide 322), where
    · *intentionality* is "something" that expresses
    · intention, design idea, purpose of artefacts –
    · well, some would say, also of natural endurants.

- As it turns out,
  - to analyse and describe mereology
  - we need to first analyse and describe unique identifiers;

  and
  - to analyse and describe attributes
  - we need to first analyse and describe mereologies.

- Hence:

Method Procedure 1 .
Sequential Analysis & Description of Internal Qualities:

- *We advise that the* domain analyser & describer
  - **first** *analyse & describe* ***unique identification*** *of all endurant sorts;*
  - **then** *analyse & describe* ***mereologies*** *of all endurant sorts;*
  - **finally** *analyse & describe* ***attributes*** *of all endurant sorts.*

5.1 Internal Qualities

- We shall investigate the, as we shall call them, internal qualities of domains.
- That is the properties of the entities to which we ascribe internal qualities.
- The outcome of this chapter is that the student
  - will be able to model the internal qualities of domains.
  - Not just for a particular domain instance,
  - but a possibly infinite set of domain instances[2].

[2] By this we mean: You are not just analysing a specific domain, say the one manifested around the corner from where you are, but any instance, anywhere in the world, which satisfies what you have described.

5.1.1 General Characterisation

- External qualities of endurants of a manifest domain
  - are, in a simplifying sense, those we can
    - see and
    - touch.
  - They, so to speak, take form.

- Internal qualities of endurants of a manifest domain
  - are, in a less simplifying sense, those which
    - we may not be able to see or "feel"
      when touching an endurant,
    - but they can, as we now 'mandate' them,
      - be reasoned about,
        as for unique identifiers
        and mereologies,

      or
    - be measured by some **physical/chemical** means,
    - or be "spoken of" by **intentional deduction**, and
      - be reasoned about,
    - as we do when we attribute properties to endurants.

5.1.2 Manifest Parts versus Structures

- In [18] we covered a notion of *'structures'*.
  - In this primer we shall treat
    the concept of 'structures' differently
  - We do so by distinguishing between
    * manifest parts
    * and structures.

5.1.2.1 Definitions

Definition 46 . ***Manifest Part:*** By a manifest part we shall understand

- a part which 'manifests' itself
  - either in a physical, visible manner, "occupying"
    an AREA or
    a VOLUME and
    a POSITION
    in SPACE,
  - or in a conceptual manner
    forms an organisation in Your mind ! ■
- As we have already revealed,
- endurant parts can be
  transcendentally deduced into perdurant behaviours
- – with manifest parts indeed being so.

Definition 47 . ***Structure:*** By a structure we shall understand

- an endurant concept that allows
  the domain analyser cum describer
  - to rationally decompose
    a domain analysis and/or its description
  - into manageable, logically relevant sections,
  - but where these abstract endurants
    are not further reflected upon
    in the domain analysis and description.
- Structures are therefore
  not transcendentally deduced into perdurant behaviours.

5.1.2.2 Analysis Predicates

Analysis Predicate Prompt 16 . *is_manifest:*

- *The method provides the* ***domain analysis prompt****:*
  - *is_manifest – where* `is_manifest(p)`
    holds if p is to be considered manifest ■

Analysis Predicate Prompt 17 . *is_structure:*

- *The method provides the* ***domain analysis prompt****:*
  - *is_structure – where* `is_structure(p)`
    holds if p is to be considered a structure ■

- The obvious holds: `is_manifest(p)` ≡ ¬ `is_structure(p)`.

5.1.2.3 Examples

Example 35 . Manifest Parts and Structures:
We refer to Example 29 on Slide 104: the Road Transport System.

- We shall consider all atomic parts: hubs, links and automobiles as being manifest. (They are physical, visible and in 𝕊ℙ𝔸ℂ𝔼.)
- We shall consider road nets and aggregates of automobiles as being manifest.
  - Road nets are physical, visible and in 𝕊ℙ𝔸ℂ𝔼.
  - Aggregates of automobiles are here considered conceptual.
  - The road net manifest part,
    - apart from it aggregates of hubs and links,
    - can be thought of as "representing" a *Department of Roads*[3].
  - The automobile aggregate
    - apart from its automobiles,
    - can be thought of as "representing" a *Department of Vehicles*[4].
  - We shall consider hub and link aggregates and hub and link set as structures.

[3] – of some country, state, province, city or other.
[4] See above footnote.

5.1.2.4 Modelling Consequence

- In this chapter we introduce internal endurant qualities.
  - If a part is considered manifest
    then we shall endow that part
    with all three kinds of internal qualities.
  - If a part is considered a structure
    then we shall **not** endow that part
    with any of three kinds of internal qualities.

5.2 Unique Identification

- The concept of parts having unique identifiability,
  - that is, that two parts,
  - if they are the same,
  - have the same unique identifier,
  - and if they are not the same,
  - then they have distinct identifiers,
  - that concept is fundamental to our being able to analyse and describe internal qualities of endurants.
- So we are left with the issue of 'identity' !

5.2.1 On Uniqueness of Endurants

- We therefore introduce the notion of unique identification of part endurants.
- We assume
  - (i) that all part endurants, e, of any domain E, have *unique identifier*s,
  - (ii) that *unique identifier*s (of part endurants e:E) are *abstract value*s (of the *unique identifier* sort UI of part endurants e:E),
  - (iii) that such that distinct part endurant sorts, E_i and E_j, have distinctly named *unique identifier* sorts, say UI_i and UI_j[5], and
  - (iv) that all ui_i:UI_i and ui_j:UI_j are distinct.

[5] This restriction is not necessary, but, for the time, we can assume that it is.

- The names of unique identifier sorts, say UI,
  is entirely at the discretion of
  the *domain analyser cum describer*.
- If, for example, the sort name of a part is P,
  then it might be expedient to
  name the sort of the unique identifiers of its parts PI.

Representation of Unique Identifiers:

- Unique identifiers are abstractions.
  - When we endow two endurants (say of the same sort) distinct unique identifiers
  - then we are simply saying that these two endurants are distinct.
  - We are not assuming anything about how these identifiers otherwise come about.

Identifiability of Endurants:

- From a philosophical point of view,
  - and with basis in Kai Sørlander's Philosphy,
  - one can rationally argue that there are many endurants,
  - and that they are unique, and hence uniquely identifiable.
- From an empirical point of view,
  - and since one may eventually have a software development in mind,
  - we may wonder how unique identifiablity can be accommodated.

- Unique identifiability for solid endurants,
  - even though they may be mobile,
  - is straightforward:
    - one can think of many ways
    - of ascribing a unique identifier to any part;
    - solid endurants do not "morph"[6].
- Hence one can think of many such unique identification schemas.

[6] That is, our domain modelling method is not thought of as being applied to the physics situations of endurants going, for example, from states of being solid, via states of melting, to states of fluid.

- Unique identifiability for fluids may seem a bit more tricky.
  - For this *primer* we shall not suggest
  - to endow fluids with unique identification.
  - We have simply not experimented with such part-fluids and fluid-parts domains – not enough – to suggest so.

5.2.2 Uniqueness Modelling Tools

- The analysis method offers an observer function uid_E
  which when applied to part endurants, e,
  yields the unique identifier, *ui*:UI, of e.

Domain Description Prompt 5 . `describe_unique_identifier(e):`

- *We can therefore apply the domain description prompt:*
  - `describe_unique_identifier(e)`
- *to endurants e:E*
  - *resulting in the analyser writing down*
  - *the unique identifier type and observer domain description text according to the following schema:*

4. `describe_unique_identifier(e)` *Observer*

❝ **Narration:**

[s] ... narrative text on unique identifier sort UI ...[7]
[u] ... narrative text on unique identifier observer uid_E ...
[a] ... axiom on uniqueness of unique identifiers ...

Formalisation:

type
[s] UI
value
[u] uid_E: E → UI ❞

- `is_part(e)` is a prerequisite for `describe_unique_identifier(e)`.

[7] The name, UI, of the unique identifier sort is determined, *"pulled out of a hat"*, by the domain analyser cum describer(s), i.e., the person(s) who "apply" the `describe_unique_identifier(e)` prompt.

- The unique identifier type name, UI above,
  - chosen, of course, by the *domain analyser cum describer*,
  - usually properly embodies the type name, E,
  - of the endurant being analysed and mereology-described.
  - Thus a part of type-name E
    might be given the mereology type name EI.
  - Generally we shall refer to these names by UI.

Observer Function Prompt 5 . `type_name, type_of, is_:`

- *Given description schema 5*
  - *we have, so-to-speak "in-reverse", that*

$$\forall\ \mathsf{e{:}E} \cdot \mathsf{uid\_E(e)=ui} \Rightarrow \mathsf{type\_of(ui)}=\eta\mathsf{UI} \wedge \mathsf{type\_name(ui)=UI} \wedge \mathsf{is\_UI(ui)}$$

- ηUI is a variable of type $\eta\mathbb{T}$.
- $\eta\mathbb{T}$ is the type of all domain endurant, unique identifier, mereology and attribute type names.
- By the subsequent UI we refer to the unique identifier type name value of ηUI.

Example 36 . Unique Identifiers:

40. We assign unique identifiers to all parts.
41. By a road identifier we shall mean a link or a hub identifier.
42. Unique identifiers uniquely identify all parts.
    (a) All hubs have distinct [unique] identifiers.
    (b) All links have distinct identifiers.
    (c) All automobiles have distinct identifiers.
    (d) All parts have distinct identifiers.

```
type
40  H_UI, L_UI, A_UI
41  R_UI = H_UI | L_UI
value
42a  uid_H: H → H_UI
42b  uid_L: H → L_UI
42c  uid_A: H → A_UI
```

5.2.3 The Unique Identifier State

- Given a universe of discourse we can calculate the set of the unique identifiers of all its parts.

```
value
  calculate_all_unique_identifiers: UoD → UI-set
  calculate_all_unique_identifiers(uod) ≡
    let parts = calc_parts({uod})({}) in
    { uid_E(e) | e:E · e ∈ parts } end
```

5.2.4 The Unique Identifier State

- We can speak of a unique identifier state:

variable
  uod := ...
  uid_σ := discover_uids()
value
  discover_uids: UoD → **Unit**
  discover_uids(uod) ≡ calculate_all_unique_identifiers(uod)

Example 37 . Unique Road Transport System Identifiers:
We can calculate:

43. the *s*et, $h_{ui}s$, of *u*nique *h*ub *i*dentifiers;
44. the *s*et, $l_{ui}s$, of *u*nique *l*ink *i*dentifiers;
45. the *s*et, $r_{ui}s$, of all *u*nique hub and link, i.e., *r*oad *i*dentifiers;
46. the *m*ap, $hl_{ui}m$, from *u*nique *h*ub *i*dentifiers
    to the *s*et of *u*nique *l*ink *i*dentifiers of the links connected
    to the zero, one or more identified hubs,
47. the *m*ap, $lh_{ui}m$, from *u*nique *l*ink *i*dentifiers
    to the *s*et of *u*nique *h*ub *i*identifiers of the two hubs connected
    to the identified link;
48. the *s*et, $a_{ui}s$, of *u*nique *a*utomobile *i*dentifiers;

value

43 $h_{ui}s$:H_UI-**set** ≡ {uid_H(h)|h:H·h ∈ hs}
44 $l_{ui}s$:L_UI-**set** ≡ {uid_L(l)|l:L·l ∈ ls}
45 $r_{ui}s$:R_UI-**set** ≡ $h_{ui}s$∪$l_{ui}s$
46 $hl_{ui}m$:(H_UI $\overrightarrow{m}$L_UI-**set**) ≡
46 [h_ui↦luis|h_ui:H_UI,luis:L_UI-**set**·h_ui∈$h_{ui}s$∧(_,luis,_)=mereo_H(η(h_ui))]
47 $lh_{ui}m$:(L+UI $\overrightarrow{m}$H_UI-**set**) ≡
47 [l_ui↦huis | h_ui:L_UI,huis:H_UI-**set** · l_ui∈$l_{ui}s$ ∧ (_,huis,_)=mereo_L(η(l_ui))]
48 $a_{ui}s$:A_UI-**set** ≡ {uid_A(a)|a:A·a ∈ as}

5.2.5 A Domain Law: Uniqueness of Endurant Identifiers

- We postulate that the unique identifier observer functions
  - are about the uniqueness of the postulated endurant identifiers,
  - but how is that guaranteed ?
  - We know, as *"an indisputable law of domains"*,
  - that they are distinct,
  - but our formulas do not guarantee that !
  - So we must formalise their uniqueness.

All Domain Parts have Unique Identifiers

A Domain Law: 1 *All Domain Parts have Unique Identifiers:*

49. All parts of a described domain have unique identifiers.

axiom
49 **card** calc_parts({uod}) = **card** all_uniq_ids()

Example 38 . Uniqueness of Road Net Identifiers:

- We must express the following axioms:

50. All hub identifiers are distinct.
51. All link identifiers are distinct.
52. All automobile identifiers are distinct.
53. All part identifiers are distinct.

axiom

50 $\quad \mathbf{card}\, hs = \mathbf{card}\, h_{ui}s$
51 $\quad \mathbf{card}\, ls = \mathbf{card}\, l_{ui}s$
52 $\quad \mathbf{card}\, as = \mathbf{card}\, a_{ui}s$
53 $\quad \mathbf{card}\, \{h_{ui}s \cup l_{ui}s \cup bc_{ui}s \cup b_{ui}s \cup a_{ui}s\}$
53 $\quad\quad = \mathbf{card}\, h_{ui}s + \mathbf{card}\, l_{ui}s + \mathbf{card}\, bc_{ui}s + \mathbf{card}\, b_{ui}s + \mathbf{card}\, a_{ui}s$ ■

- We ascribe, in principle, unique identifiers
  - to all endurants
    - whether natural
    - or artefactual.
- We find, from our many experiments,
  cf. the *Universes of Discourse* example, Page 52,
  - that we really focus on those domain entities which are
    - artefactual endurants and
    - their behavioural "counterparts".

Example 39 . Rail Net Unique Identifiers:

54. With every rail net unit we associate a unique identifier.
55. That is, no two rail net units have the same unique identifier.
56. Trains have unique identifiers.
57. We let *tris* denote the set of all train identifiers.
58. No two distinct trains have the same unique identifier.
59. Train identifiers are distinct from rail net unit identifiers.

type
54. UI
value
54. uid_NU: NU → UI
axiom
55. ∀ ui_i,ui_j:UI · ui_i = ui_j ≡ uid_NU(ui_i)=uid_NU(ui_j)

5.2.5.1 Part Retrieval

- Given the unique identifier, pi, of a part p,
- but not the part itself,
- and given the universe-of-discourse (uod) state σ,
- we can *retrieve* part, p, as follows:

value
  pi:PI, uod:UoD, σ
  retr_part: UI → P
  retr_part(ui) ≡ **let** p:P · p ∈ σ ∧ uid_P(p)=ui **in** p **end**
    pre: ∃ p:P · p ∈ σ ∧ uid_P(p)=ui

5.2.5.2 Unique Identification of Compounds

- For structures we do not model their unique identification.
  - But their components,
    - whether the structures are *"Cartesian"*
    - or *"sets"*,
  - may very well be non-structures, hence be uniquely identifiable.

5.3 Mereology

Definition 48 . ***Mereology:*** Mereology is the study and knowledge of parts and part relations ■

- Mereology, as a logical/philosophical discipline,
  can perhaps best be attributed to
  the Polish mathematician/logician
  Stanisław Leśniewski [26, 9].

5.3.1 Endurant Relations

- Which are the relations
  that can be relevant for "endurant-hood" ?
- There are basically two relations:
  - (i) physical ones, and
  - (ii) conceptual ones.
- (i) Physically two or more endurants may be topologically
  - either adjacent to one another, like rails of a line,
  - or within an endurant, like links and hubs of a road net,
  - or an atomic part is conjoined to one or more fluids,
  - or a fluid is conjoined to one or more parts.

- The latter two could also be considered conceptual “adjacencies”.
- (ii) Conceptually some parts, like automobiles,
  - “belong” to an embedding endurant,
    - like to an automobile club, or
    - are registered in the local department of vehicles,
  - or are ‘intended’ to drive on roads.

5.3.2 Mereology Modelling Tools

- When the domain analyser decides that
  - some endurants are related in a specifically enunciated mereology,
  - the analyser has to decide on suitable
    - ***mereology type***s and
    - ***mereology observer***s (i.e., endurant relations).

60. We may, to illustration, define a ***mereology type*** of an endurant $e{:}E$ as a triplet type expression over set of unique [endurant] identifiers.
61. There is the identification of all those endurant sorts $E_{i_1}, E_{i_2}, ..., E_{i_m}$ where at least one of whose properties "`is_of_interest`" to parts $e{:}E$.
62. There is the identification of all those sorts $E_{io_1}, E_{io_2}, ..., E_{io_n}$ where at least one of whose properties "`is_of_interest`" to endurants $e{:}E$ and vice-versa.
63. There is the identification of all those endurant sorts $E_{o_1}, E_{o_2}, ..., E_{o_o}$ for whom properties of $e{:}E$ "`is_of_interest`" to endurants of sorts $E_{o_1}, E_{o_2}, ..., E_{o_o}$.
64. The mereology triplet sets of unique identifiers are disjoint and are all unique identifiers of the universe of discourse.

- The triplet mereology is just a suggestion.
  - As it is formulated here
    we mean the three ‘sets’ to be disjoint.
  - Other forms of expressing a mereology
    should be considered
  - for the particular domain
    and for the particular endurants of that domain.
- We leave out further characterisation of
  - the seemingly vague notion "`is_of_interest`".

```
type
61  iEI  = iEI1 | iEI2 | ... | iEIm
62  ioEI = ioEI1 | ioEI2 | ... | ioEIn
63  oEI  = oEI1 | oEI2 | ... | oEIo
60  MT = iEI-set × ioEI-set × oEI-set
axiom
64  ∀ (iset,ioset,oset):MT ·
64    card iset + card ioset + card oset = card ∪{iset,ioset,oset}
64    ∪{iset,ioset,oset} ⊆ calc_all_unique_identifiers(uod)
```

Domain Description Prompt 6 . `describe_mereology(e)`:

- *If* `has_mereology(p)` *holds for parts* p *of type* P,
  - *then the analyser can apply the* **domain description prompt**:
    - `describe_mereology`
  - *to parts of that type*
  - *and write down the* mereology types and observer *domain description text according to the following schema:*

5. `describe_mereology(e)` Observer

❝ **Narration:**

[t] ... narrative text on mereology type ...
[m] ... narrative text on mereology observer ...
[a] ... narrative text on mereology type constraints ...

Formalisation:

type
[t] MT = $\mathcal{M}$(UI$_i$,UI$_j$,...,UI$_k$)
value
[m] mereo_P: P → MT
axiom [Well–formedness of Domain Mereologies]
[a] $\mathcal{A}$: $\mathcal{A}$(MT) ❞

- The mereology type name, MT,
  chosen of course, by the *domain analyser cum describer*,
  usually properly embodies the type name, E,
  of the endurant being analysed and mereology-described.
- The mereology type expression $\mathcal{M}(\mathsf{UI}_i,\mathsf{UI}_j,...,\mathsf{UI}_k)$
  is a type expression over unique identifiers.
  - Thus a part of type-name P
    might be given the mereology type name MP.
- $\mathcal{A}$(MT) is a predicate
  over possibly all unique identifier types
  of the domain description.
- To write down the concrete type definition for MT
  requires a bit of analysis and thinking ■

Example 40 . Mereology of a Road Net:

65. The mereology of hubs is a pair:
    (i) the set of all automobile identifiers[8], and
    (ii) the set of unique identifiers of the links that it is connected to and the set of all unique identifiers of all automobiles.[9]
66. The mereology of links is a pair:
    (i) the set of all bus and automobile identifiers, and
    (ii) the set of the two distinct hubs they are connected to.
67. The mereology of an automobile is
    the set of the unique identifiers of all links and hubs[10].

- We presently omit treatment of road net and automobile aggregate mereologies.
- For road net mereology we refer to Example 69, Item 153 on Slide 418.

```
type
65  H_Mer = V_UI-set×L_UI-set
66  L_Mer = V_UI-set×H_UI-set
67  A_Mer = R_UI-set
value
65  mereo_H: H → H_Mer
66  mereo_L: L → L_Mer
67  mereo_A: A → A_Mer
```

[7] This is just another way of saying that the meaning of hub mereologies involves the unique identifiers of all the vehicles that might pass through the hub `is_of_interest` to it.

[8] The link identifiers designate the links, zero, one or more, that a hub is connected to `is_of_interest` to both the hub and that these links is `interested` in the hub.

[9] — that the automobile might pass through

5.3.2.1 Invariance of Mereologies

- For mereologies one can usually express some invariants.
  - Such invariants express *"law-like properties"*,
  - facts which are indisputable.

Example 41 . Invariance of Road Nets:

- The observed mereologies
  must express identifiers of the state of such for road nets:

axiom

65 ∀ (auis,luis):H_Mer · luis⊆$l_{ui}s$ ∧ auis=$a_{ui}s$

66 ∀ (auis,huis):L_Mer · auis=$a_{ui}s$ ∧ huis⊆$h_{ui}s$ ∧ **card** huis = 2

67 ∀ ruis:A_Mer · ruis=$r_{ui}s$

68. For all hubs, h, and links, l, in the same road net,
69. if the hub h connects to link l
    then link l connects to hub h.

axiom

68 ∀ h:H,l:L · h ∈ hs ∧ l ∈ ls ⇒

68 **let** (_,luis)=mereo_H(h), (_,huis)=mereo_L(l)

69 **in** uid_L(l)∈luis ≡ uid_H(h)∈huis **end**

70. For all links, l, and hubs, h_a, h_b, in the same road net,
71. if the l connects to hubs h_a and h_b,
    then h_a and h_b both connects to link l.

axiom
70 ∀ h_a,h_b:H,l:L · {h_a,h_b} ⊆ *hs* ∧ l ∈ *ls* ⇒
70 **let** (_,luis)=mereo_H(h), (_,huis)=mereo_L(l)
71 **in** uid_L(l)∈luis ≡ uid_H(h)∈huis **end**

5.3.2.2 Deductions made from Mereologies

- Once we have settled basic properties of the mereologies of a domain
  - we can, like for unique identifiers, cf. Example 36 on Slide 189,
  - *"play around"* with that concept:
    'the mereology of a domain'.

Example 42 . Consequences of a Road Net Mereology:

72. are there [isolated] units from which one
    can not "reach" other units ?
73. does the net consist of two or more "disjoint" nets ?
74. et cetera.

- We leave it to the reader to
  narrate and formalise the above properly.

5.3.3 Formulation of Mereologies

- The `observe_mereology` domain descriptor, Slide 210,
  - may give the impression that
    the mereo type MT can be described
  - "at the point of issue" of the `observe_mereology` prompt.
  - Since the MT type expression may depend on any part sort
  - the mereo type MT can, for some domains,
  - "first" be described
    when all part sorts
    have had their unique identifiers defined.

5.3.4 Fixed and Varying Mereologies

- The mereology of parts is not necessarily fixed.

Definition 49 . *Fixed Mereology:*

- By a **fixed mereology** we shall understand
  - a mereology of a part
  - which remains fixed
  - over time.

Definition 50 . *Varying Mereology:*

- By a **varying mereology** we shall understand
  - a mereology of a part
  - which may vary
  - over time.

Example 43 . Fixed and Varying Mereology:

- Let us consider a road net[10].
  - If hubs and links never change "affiliation", that is:
    * hubs are in fixed relation to zero one or more links, and
    * links are in a fixed relation to exactly two hubs
    * then the mereology of
      Example 40 on Slide 212 is a *fixed mereology*.

[10] cf. Examples 21 on Slide 54,
29 on Slide 104,
30 on Slide 117,
32 on Slide 127,
35 on Slide 176,
36 on Slide 189,
38 on Slide 197,
39 on Slide 199,
40 on Slide 212 and
41 on Slide 215

– If, on the other hand
  * hubs may be inserted into or removed from the net,
    and/or
  * links may be removed from
    or inserted between any two existing hubs,
  * then the mereology of Example 40 on Slide 212
    is a *varying mereology*.

5.3.5 No Fluids Mereology

- We comment on our decision, for this *primer*, to not endow fluids with mereologies.
  - A first reason is that we "restrict" the concept of mereology to part endurants, that is, to solid endurants – those with "more-or-less" *fixed extents*.
  - Fluids can be said to normally not have fixed extents, that is, they can "morph" from small, fixed into spatially extended forms.

– For domains of part-fluid conjoins this is particularly true.
– The fluids in such domains flow through and between parts.
– Some parts, at some times, embodying large,
  at other times small amounts of fluid.
– Some proper, but partial amount of fluid
  flowing from one part to a next.
– Et cetera.
– It is for the same reason that
  we do not endow fluids with identity.

• So, for this *primer* we decide to not suggest the modelling of fluid mereologies.

5.3.6 Some Modelling Observations

- It is, in principle, possible to find examples of mereologies of natural parts:
  - rivers: their confluence, lakes and oceans; and
  - geography: mountain ranges, flat lands, etc.
- But in our experimental case studies, cf. Example on Page 52, we have found no really interesting such cases.
- All our experimental case studies appears to focus on the mereology of artefacts.

- And, finally, in modelling humans,
  - we find that their mereology encompass
    - all other humans
    - and all artefacts !
  - Humans cannot be tamed to refrain from interacting with everyone and everything.
- Some domain models may emphasize *physical mereologies* based on spatial relations,
- others may emphasize *conceptual mereologies* based on logical "connections".

- Some domain models may emphasize *physical mereologies* based on spatial relations,
- others may emphasize *conceptual mereologies* based on logical "connections".

Example 44 . Rail Net Mereology:

75. A linear rail unit is connected to exactly
    two distinct other rail net units of any given rail net.
76. A point unit is connected to exactly
    three distinct other rail net units of any given rail net.
77. A rigid crossing unit is connected to exactly
    four distinct other rail net units of any given rail net.
78. A single and a double slip unit is connected to exactly
    four distinct other rail net units of any given rail net.
79. A terminal unit is connected to exactly
    one distinct other rail net unit of any given rail net.
80. So we model the mereology of a railway net unit
    as a pair of sets of rail net unit unique identifiers
    distinct from that of the rail net unit.

```
value
80.  mereo_NU: NU → (UI-set×UI-set)
axiom
80.  ∀ nu:NU ·
80.    let (uis_i,uis_o)=mereo_NU(nu) in
80.    case (card uis_i,card usi_o) =
75.       (is_LU(nu) → (1,1),
76.        is_PU(nu) → (1,2) ∨ (2,1),
77.        is_RU(nu)  → (2,2),
78.        is_SU(nu) → (2,2), is_DU(nu) → (2,2),
79.        is_TU(nu)  → (1,0) ∨ (0,1),
80.        _ → chaos) end
80.  ∧ uis_i∩uis_o={}
80.  ∧ uid_NU(nu) ∉ (uis_i ∪ uis_o)
80.   end
```

- Figure 5.1
  - illustrates the mereology of four rail units.

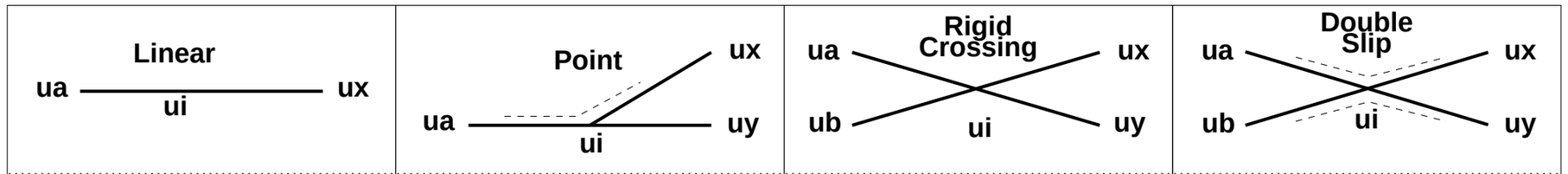

Figure 5.1: Four Symmetric Rail Unit Mereologies

Lecture 4: Attributes and Summary

5.4 Attributes

- To recall: there are three sets of ***internal qualities***:
  - unique identifiers,
  - mereologies and
  - attributes.
- Unique identifiers and mereologies
  are rather definite kinds of internal endurant qualities;
- attributes form more "free-wheeling" sets of ***internal qualities***.
- Whereas, for this *primer*, we suggest to not endow
  fluids with unique identification and mereologies
  all endurants, i.e., including fluids, are endowed with attributes.

5.4.1 Inseparability of Attributes from Parts and Fluids

- Parts and fluids are
  - typically recognised because of their spatial form
  - and are otherwise characterised by their intangible, but measurable attributes.
- We equate all endurants — which have

  the same type of unique identifiers,

  the same type of mereologies,

  and the same types of attributes

  — with one sort.
- Thus removing an internal quality from an endurant makes no sense:
  - the endurant of that type
  - either becomes an endurant of another type
  - or ceases to exist (i.e., becomes a non-entity) !

- We can roughly distinguish between two kinds of attributes:
  - those which can be motivated
    by **physical** (incl. chemical) **concerns**, and
  - those,
    * which, although they embody some form of 'physics measures',
    * appear to reflect on **event histories**:
      · *"if 'something', ϕ, has 'happened' to an endurant, e_a,*
      · *then some 'commensurate thing', ψ, has 'happened' to another (one or more) endurants, e_b."*
    * where the *'something'* and *'commensurate thing'*
    * usually involve some 'interaction' between the two (or more) endurants.
  - It can take some reflection and analysis to properly identify
    * endurants e_a and e_b and
    * commensurate events ϕ and ψ.

5.4.2 Attribute Modelling Tools

5.4.2.1 Attribute Quality and Attribute Value

- We distinguish between
  - an **attribute** (as a logical proposition, of a name, i.e.) **type**, and
  - an **attribute value**, as a value in some value space.

5.4.2.2 Concrete Attribute Types

- By a *concrete type* shall understand a sort (i.e., a type) which is defined in terms of some type expression: $\mathsf{T} = \mathcal{T}(...)$.
- This is referred to below as [=...].

5.4.2.3 Attribute Types and Functions

- Let us recall that attributes cover qualities other than unique identifiers and mereology.
- Let us then consider that parts and fluids to have one or more attributes.
  - These attributes are qualities
  - which help characterise "what it means" to be a part or a fluid.
- Note that we expect every part and fluid to have at least one attribute.
- The question is now, in general, how many and, particularly, which.

Domain Description Prompt 7 . describe_attributes:

- *The domain analyser experiments, thinks and reflects about endurant,* e, *attributes.*
- *That process is initiated by the* domain description prompt*:*
  - describe_attributes(e).
- *The result of that* domain description prompt *is that the domain analyser cum describer writes down the* attribute (sorts or) types and observers *domain description text according to the following schema:*

6. `describe_attributes` Observer

let $\{\eta A_1, ..., \eta A_m\}$ = analyse_attribute_type_names(e) **in**
❝ **Narration:**
[t] ... narrative text on attribute sorts ...
some Ais may be concretely defined: [Ai=...]
[o] ... narrative text on attribute sort observers ...
[p] ... narrative text on attribute sort proof obligations ...
Formalisation:
type
[t] A_1[=...] , ..., A_m[=...]
value
[o] attr_A_1: E→A_1, ..., attr_A_m: E→A_m
proof obligation [Disjointness of Attribute Types]
[p] $\mathcal{PO}$: **let** P be any part sort **in** [the domain description]
[p] **let** a:(A_1|A_2|...|A_m) **in** is_A_i(a) ≠ is_A_j(a) [i≠i, i,j:[1..m]] **end end** ❞
end

- Let A_1, ..., A_n be the set of all conceivable attributes of endurants $e{:}E$.
  - (Usually n is a rather large natural number, say in the order of a hundred conceivable such.)
  - In any one domain model the domain analyser cum describer selects a modest subset, A_1, ..., A_m, i.e., $m < n$.
  - Across many domain models
    for *"more-or-less the same"* domain m varies
    and the attributes, A_1, ..., A_m,
    selected for one model may differ
    from those, A'_1, ..., $A'_{m'}$, chosen for another model.

- The **type** definitions: A_1, ..., A_m, inform us that the domain analyser has decided to focus on the distinctly named A_1, ..., A_m attributes.
- The **value** clauses
  - attr_A_1:P→A_1,
  - ...,
  - attr_A_n:P→A_n

  are then "automatically" given:
  - if an endurant, e:E, has an attribute A_i
  - then there is postulated, "by definition" [eureka] an attribute observer function attr_A_i:E→A_i et cetera ■

- We cannot automatically, that is, syntactically, guarantee that our domain descriptions secure that
  - the various attribute types
  - for a endurant sort
  - denote disjoint sets of values.

  Therefore we must prove it.

5.4.2.4 Attribute Categories

- Michael A. Jackson [33] has suggested a hierarchy of attribute categories:
  - from static
  - to dynamic values – and within the dynamic value category:
    - inert values,
    - reactive values,
    - active values – and within the dynamic active value category:
      - autonomous values,
      - biddable values and
      - programmable values.
- We now review these attribute value types. The review is based on [33, M.A.Jackson].

- *Endurant attributes* are
  - either constant, i.e., static,
  - or varying, i.e., dynamic

  attributes

Attribute Category: **1** .

- By a *static attribute*, a:A, `is_static_attribute`(a), we shall understand an attribute whose values
  - are constants,
  - i.e., cannot change ■

Example 45 . Static Attributes:

- Let us exemplify road net attributes
  in this and the next examples.
- And let us assume the following attributes:
  - year of first link construction and
  - link length at that time.
- We may consider both to be static attributes:
  - The year first established,
    seems an obvious static attribute and
  - the length is fixed at the time the road was first built.

Attribute Category: 2 .

- By a *dynamic attribute*, a:A, `is_dynamic_attribute`(a),
  we shall understand an attribute whose values
  - are variable,
  - i.e., can change.

  Dynamic attributes are either *inert, reactive* or *active* attributes ■

Attribute Category: 3 .

- By an *inert attribute*, a:A, `is_inert_attribute`(a),
  we shall understand a dynamic attribute whose values
  - only change as the result of external stimuli where
  - these stimuli prescribe new values ■

Example 46 . Inert Attribute:

- And let us now further assume the following link attribute:
  - link name.
- We may consider it to be an inert attribute:
  - the name is not "assigned" to the link by the link itself,
  - but probably by some road net authority
  - which we are not modelling.

Attribute Category: 4 .

- By a ***reactive attribute***, a:A, `is_reactive_attribute`(a), we shall understand a dynamic attribute whose values,
  - if they vary, change in response to external stimuli,
  - where these stimuli
    - either come from outside the domain of interest
    - or from other endurants ■

Example 47 . Reactive Attributes:

- Let us further assume the following two link attributes:
  - "wear and tear", respectively
  - "icy and slippery".
- We will consider those attributes to be reactive in that
  - automobiles (another part) traveling the link, an external "force",
    typically causes the "wear and tear", respectively
  - the weather (outside our domain)
    causes the "icy and slippery" property.

Attribute Category: 5 .

- By an ***active attribute***, a:A, `is_active_attribute`(a), we shall understand a dynamic attribute whose values
  - change (also) of its own volition.

  Active attributes are
  - either *autonomous*,
  - or *biddable*
  - or *programmable*

  attributes ■

Attribute Category: **6** .

- By an *autonomous attribute*, a:A, `is_autonomous_attribute`(a), we shall understand a dynamic active attribute
  - whose values change only "on their own volition".
  - The values of an autonomous attributes are a "law onto themselves and their surroundings" ■

Example 48 . Autonomous Attributes:

- We enlarge scope of our examples of attribute categories to now also include automobiles (on the road net).
- In this example we assume that an automobile is driven by a human [behaviour].
- These are some automobile attributes:
  - velocity,
  - acceleration, and
  - moving straight, or turning left, or turning right.
- We shall consider these three attributes to be autonomous.
  - It is the driver, not the automobile, who decides
  - whether the automobile should drive at constant velocity, including 0, or accelerate or decelerate, including stopping.
  - And it is the driver who decides when to turn left or right, or not turn at all.

Attribute Category: 7 .

- By a *biddable attribute*, a:A, `is_biddable_attribute`(a) we shall understand a dynamic active attribute whose values
  - *are prescribed*
  - *but may fail to be observed as such* ■

Example 49 . Biddable Attributes: In the context of automobiles these are some biddable attributes:

- turning the wheel, to drive right at a hub
  – with the automobile failing to turn right;
- pressing the accelerator, to obtain a higher speed
  – with the automobile failing to really gaining speed;
- pressing the brake, to stop
  – with the automobile failing to halt ■

Attribute Category: 8 .

- By a *programmable attribute*, a:A, `is_programmable_attribute`(a), we shall understand a dynamic active attribute whose values
  - can be prescribed ■

Example 50 . Programmable Attribute:

- We continue with the automobile on the road net examples.
- In this example we assume that an automobile includes, as one inseparable entity, “the driver”.
- These are some automobile attributes:
  - position on a link,
  - velocity, acceleration (incl. deceleration), and
  - direction: straight, turning left, turning right.
- We shall now consider these three attributes to be programmable.

- Figure 5.2 captures an attribute value ontology.

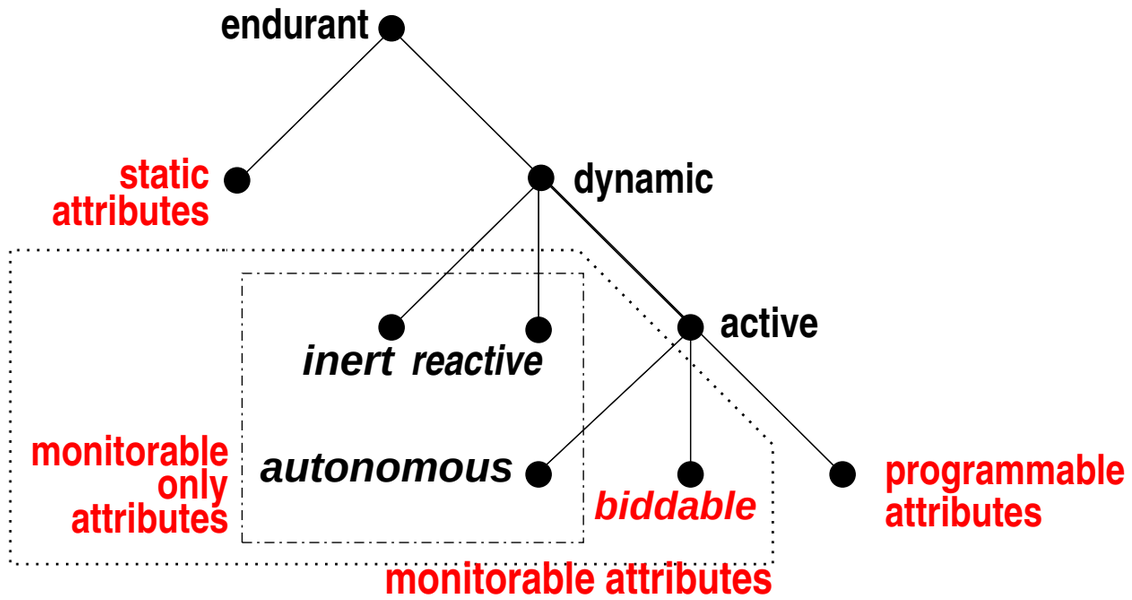

Figure 5.2: Attribute Value Ontology

- Figure 5.2 hints at three categories of dynamic attributes:
  - monitorable only,
  - biddable and
  - programmable

  attributes.

Attribute Category: 9 .

- By a *monitorable only attribute*, a:A,
  `is_monitorable_only_attribute`(a),
  we shall understand a dynamic active attribute which is either
  - *inert* or
  - *reactive* or
  - *autonomous*.

That is:

```
value
  is_monitorable_only: E → Bool
  is_monitorable_only(e) ≡ is_inert(e) ∨ is_reactive(e) ∨ is_autonomous(e)
```

Example 51 . Road Net Attributes:

- We treat some attributes of the hubs of a road net.

81. There is a hub state.
    - It is a set of pairs, (l_f,l_t), of link identifiers,
      – where these link identifiers are in the mereology of the hub.
    - The meaning of the hub state
      – in which, e.g., (l_f,l_t) is an element,
      – is that the hub is open, "green",
      – for traffic *f*rom link l_f *t*o link l_t.
      – If a hub state is empty
      – then the hub is closed, i.e., "red"
      – for traffic from any connected links to any other connected links.

82. There is a hub state space.
    - It is a set of hub states.
    - The current hub state must be in its state space.
    - The meaning of the hub state space is
      – that its states are all those the hub can attain.

83. Since we can think rationally about it,
    - it can be described, hence we can model, as an attribute of hubs, a history of its traffic:
      – the recording, per unique bus and automobile identifier,
      – of the time ordered presence in the hub of these vehicles.
    - Hub history is an *event history*.

type
81 HΣ = (L_UI×L_UI)-**set**
82 HΩ = HΣ-**set**
83 H_Traffic = (A_UI|B_UI) $\xrightarrow[m]{}$ ($\mathbb{TIME}$ × VPos)*
axiom
81 ∀ h:H · obs_HΣ(h) ∈ obs_HΩ(h)
83 ∀ ht:H_Traffic,ui:(A_UI|B_UI) · ui ∈ **dom** ht ⇒ time_ordered(ht(ui))
value
81 attr_HΣ: H → HΣ
82 attr_HΩ: H → HΩ
83 attr_H_Traffic: H → H_Traffic
83 time_ordered: ($\mathbb{TIME}$ × VPos)* → **Bool**
83 time_ordered(tvpl) ≡ ...

- In Item 83 we model the time-ordered sequence of traffic as a discrete sampling, i.e., $\xrightarrow[m]{}$, rather than as a continuous function, →.

Example 52 . Invariance of Road Net Traffic States:

- We continue Example 51 on Slide 258.

84. The link identifiers of hub states must be in the set, $l_{ui}s$, of the road net's link identifiers.

axiom
84 ∀ h:H · h ∈ hs ⇒
84 **let** hσ = attr_HΣ(h) **in**
84 ∀ ($l_{ui}i$,$li_{ui}i'$):(L_UI×L_UI) · ($l_{ui}i$,$l_{ui}i'$) ∈ hσ ⇒ {l_{ui_i},l'_{ui_i}} ⊆ $l_{ui}s$ **end**

- You may skip Example 53 in a first reading.

Example 53 . Road Transport: Further Attributes:
Links:
We show just a few attributes.

85. There is a link state. It is a set of pairs, (h_f,h_t), of distinct hub identifiers, where these hub identifiers are in the mereology of the link. The meaning of a link state in which (h_f,h_t) is an element is that the link is open, "green", for traffic f rom hub h_f to hub h_t. Link states can have either 0, 1 or 2 elements.
86. There is a link state space. It is a set of link states. The meaning of the link state space is that its states are all those the which the link can attain. The current link state must be in its state space. If a link state space is empty then the link is (permanently) closed. If it has one element then it is a one-way link. If a one-way link, l, is imminent on a hub whose mereology designates that link, then the link is a "trap", i.e., a "blind cul-de-sac".

87. Since we can think rationally about it, it can be described, hence it can model, as an attribute of links a history of its traffic: the recording, per unique bus and automobile identifier, of the time ordered positions along the link (from one hub to the next) of these vehicles.
88. The hub identifiers of link states must be in the set, $h_{ui}s$, of the road net's hub identifiers.

type
85 LΣ = H_UI-**set**
86 LΩ = LΣ-**set**
87 L_Traffic
87 L_Traffic = (A_UI|B_UI) $\xrightarrow[m]{}$ ($\mathbb{T}$×(H_UI×Frac×H_UI))*
87 Frac = **Real**, **axiom** frac:Fract · 0<frac<1
value
85 attr_LΣ: L → LΣ
86 attr_LΩ: L → LΩ
87 attr_L_Traffic: : → L_Traffic
axiom
85 ∀ lσ:LΣ·**card** lσ=2
85 ∀ l:L · obs_LΣ(l) ∈ obs_LΩ(l)
87 ∀ lt:L_Traffic,ui:(A_UI|B_UI)·ui ∈ **dom** ht ⇒ time_ordered(ht(ui))
88 ∀ l:L · l ∈ *ls* ⇒ **let** lσ = attr_LΣ(l) **in** ∀ (h$_{ui}i$,h$_{ui}i'$):(H_UI×K_UI) ·
88 (h$_{ui}i$,h$_{ui}i'$) ∈ lσ ⇒ {h$_{ui_i}$,h$'_{ui_i}$} ⊆ h$_{ui}s$ **end**

Automobiles:

- We illustrate but a few attributes:

89. Automobiles have static number plate registration numbers.
90. Automobiles have dynamic positions on the road net:
    (a) either *at a hub* identified by some h_ui,
    (b) or *on a link*, some *fraction, frac:Fract* down an *identified link, l_ui*, from one of its *identified connecting hubs*, fh_ui, in the direction of the other *identified hub*, th_ui.
    (c) Fraction is a real properly between 0 and 1.

type
89 RegNo
90 APos == atHub | onLink
90a atHub :: h_ui:H_UI
90b onLink :: fh_ui:H_UI × l_ui:L_UI × frac:Fract × th_ui:H_UI
90c Fract = **Real**
axiom
90c frac:Fract · 0<frac<1
value
89 attr_RegNo: A → RegNo
90 attr_APos: A → APos

- Obvious attributes that are not illustrated are those of
  - velocity and acceleration,
  - forward or backward movement,
  - turning right, left or going straight,
  - etc.

- The *acceleration, deceleration, even velocity,* or *turning right, turning left, moving straight*, or *forward* or *backward* are seen as *command actions*.
  - As such they denote actions by the automobile —
  - such as pressing the accelerator, or lifting accelerator pressure or *braking*, or *turning the wheel* in one direction or another, etc.
  - As actions they have a kind of counterpart in the velocity, the acceleration, etc. attributes.

- In Items Sli. 259 and Sli. 263, we illustrated an aspect of domain analysis & description that may seem, and at least some decades ago would have seemed, strange: namely that if we can think, hence speak, about it, then we can model it "as a fact" in the domain. The case in point is that we include among hub and link attributes their histories of the timed whereabouts of buses and automobiles[11] ■

[11] In this day and age of road cameras and satellite surveillance these traffic recordings may not appear so strange: We now know, at least in principle, of technologies that can record approximations to the hub and link traffic attributes.

5.4.2.5 Calculating Attribute Category Type Names

- One can calculate sets of all attribute type names, of static, so-called monitorable and programmable attribute types of parts and fluids with the following ***domain analysis prompt***s:
  - `analyse_attribute_type_names`,
  - `sta_attr_types`,
  - `mon_attr_types`, and
  - `pro_attr_types`.
    - `analyse_attribute_type_names` applies to parts and yields a set of all attribute names of that part.
    - `mon_attr_types` applies to parts and yields a set of attribute names of *monitorable* attributes of that part.[12]

[12] $\eta\mathbb{A}$ is the type of all attribute types.

Observer Function Prompt 6 . *analyse_attribute_types:*

value
analyse_attribute_type_names: P → ηA-**set**
analyse_attribute_type_names(p) **as** {ηA1,ηA,...,ηAm }

Observer Function Prompt 7 . *sta_attr_types:*

value
  sta_attr_types: P → $\eta\mathbb{A}\times\eta\mathbb{A}\times...\times\eta\mathbb{A}$
  sta_attr_types(p) **as** (ηA1,ηA2,...,ηAn)
    where: {ηA1,ηA2,...,ηAn} ⊆ analyse_attribute_type_names(p)
      ∧ **let** anms = analyse_attribute_type_names(p)
        ∀ anm:$\eta\mathbb{A}$ · anm ∈ anms \ {ηA1,ηA2,...,ηAn}
          ⇒ ~ is_static_attribute{anm}
      ∧ ∀ anm:$\eta\mathbb{A}$ · anm ∈ {ηA1,ηA2,...,ηAn}
        ⇒ is_static_attribute{anm} **end**

Observer Function Prompt 8 . *mon_attr_types:*

```
value
  mon_attr_types: P → ηA×ηA×...×ηA
  mon_attr_types(p) as (ηA1,ηA2,...,ηAn)
    where: {ηA1,ηA2,...,ηAn} ⊆ analyse_attribute_type_names(p)
      ∧ let anms = analyse_attribute_type_names(p)
        ∀ anm:ηA · anm ∈ anms \ {ηA1,ηA2,...,ηAn}
          ⇒ ~ is_monitorable_attribute{anm}
      ∧ ∀ anm:ηA · anm ∈ {ηA1,ηA2,...,ηAn}
        ⇒ is_monitorable_attribute{anm} end
```

Observer Function Prompt 9 . *pro_attr_types:*

value
  pro_attr_types: P → $\eta\mathbb{A}\times\eta\mathbb{A}\times...\times\eta\mathbb{A}$
  pro_attr_types(p) **as** (ηA1,ηA2,...,ηAn)
    where: {ηA1,ηA2,...,ηAn} ⊆ analyse_attribute_type_names(p)
      ∧ **let** anms = analyse_attribute_type_names(p)
        ∀ anm:$\eta\mathbb{A}$ · anm ∈ anms \ {ηA1,ηA2,...,ηAn}
          ⇒ ~ is_monitorable_attribute{anm}
      ∧ ∀ anm:$\eta\mathbb{A}$ · anm ∈ {ηA1,ηA2,...,ηAn}
        ⇒ is_monitorable_attribute{anm} **end**

- Some comments are in order.
  - The analyse_attribute_type_names function is, as throughout, meta-linguistic, that is, informal, not-computable, but decidable by the domain analyser cum describer. Applying it to a part or fluid yields, at the discretion of the domain analyser cum describer, a set of attribute type names "freely" chosen by the domain analyser cum describer.
  - The sta_attr_type_names, the mon_attr_type_names, and the pro_attr_type_names functions are likewise meta-linguistic; their definition here relies on the likewise meta-linguistic is_static, is_monitorable and is_programmable analysis predicates.

5.4.2.6 Calculating Attribute Values

- Let (ηA1, ηA2, ... , ηAn) be a grouping of attribute types for part p (or fluid f).
- Then (attr_A1(p), attr_A2(p), ... ,attr_An(p))
- (respectively f)
- yields (a1, a2, ... , an), the grouping of values for these attribute types.
- We can "formalise" this conversion:

value
  types_to_values: $\eta\mathbb{A}_1 \times \eta\mathbb{A}_2 \times ... \times \eta\mathbb{A}_n \rightarrow \mathsf{A}_1 \times \mathsf{A}_2 \times ... \times \mathsf{A}_n$

5.4.2.7 Calculating Attribute Names

- The meta-linguistic, i.e., "outside" RSL proper, name for attribute type names is introduced here as $\eta\mathbb{A}$.

91. Given endurant *e* we can *meta-linguistically*[13] calculate names for its *static* attributes.
92. Given endurant *e* we can *meta-linguistically* calculate name for its *monitorable* attributes attributes.
93. Given endurant *e* we can *meta-linguistically* calculate names for its *programmable* attributes.
94. These four sets make up all the attributes of endurant *e*.

[13] By using the term *meta-linguistically* here we shall indicate that we go outside what is computable – and thus appeal to the reader's forbearance.

The type names ST, MA, PT designate mutually disjoint

- sets, ST, of names of static attributes,
- sets, MA, of names of monitoriable, i.e., monitorable-only and biddable, attributes,
- sets, PT, of names of programmable, i.e., fully controllable attributes.

type
91 ST = $\eta\mathbb{A}$-**set**
92 MA = $\eta\mathbb{A}$-**set**
93 PT = $\eta\mathbb{A}$-**set**
value
91 stat_attr_types: E → ST
92 moni_attr_types: E → MA
93 prgr_attr_types: E → PT

axiom
94 ∀ e:E ·
91 **let** stat_nms = stat_attr_types(e),
92 moni_nms = moni_attr_types(e),
93 prgr_nms = prgr_types(e) **in**
94 **card** stat_nms + **card** moni_nms + **card** prgr_nms
94 = **card**(stat_nms ∪ mon_nms ∪ prgr_nms) **end**

The above formulas are indicative, like mathematical formulas, they are not computable.

95. Given endurant *e* we can *meta-linguistically* calculate its static attribute values, stat_attr_vals;
96. given endurant *e* we can *meta-linguistically* calculate its monitorable-only attribute values, moni_attr_vals; and
97. given endurant *e* we can *meta-linguistically* calculate its programmable attribute values, prgr_attr_vals.

The type names sa1, ..., pap refer to the types denoted by the corresponding types name nsa1, ..., npap.

```
value
95  stat_attr_vals: E → SA1×SA2×...×SAs
95  stat_attr_vals(e) ≡
95    let {nsa1,nsa2,...,nsas} = stat_attr_types(e) in
95    (attr_sa1(e),attr_sa2(e),...,attr_sas(e)) end

96  moni_attr_vals: E → MA1×MA2×...×MAm
96  moni_attr_vals(e) ≡
96    let {nma1,nma2,...,nmam} = moni_attr_types(e) in
96    (attr_ma1(e),attr_ma2(e),...,attr_mam(e)) end
```

```
97  prgr_attr_vals: E → PA1×PA2×...×PAp
97  prgr_attr_vals(e) ≡
97    let {npa1,npa2,...,npap} = prgr_attr_types(e) in
97    (attr_pa1(e),attr_pa2(e),...,attr_pap(e)) end
```

- The "ordering" of type values,
  - (attr_sa1(e),...,attr_sas(e)),
  - (attr_ma1(e),...,attr_mam(e)), et cetera,
  - is arbitrary.

5.4.3 Operations on Monitorable Attributes of Parts

- We remind the student of the notions of
  - states in general, Sect. 4.7 and
  - updateable states, Sect. 4.7.2 on Slide 153.
    - For every domain description there possibly is an updateable state.
    - The is such a state if there is at least one part with at least one monitorable attribute.
  - Below we refer to the updateable states as σ.

- Given a part, p, with attribute A,
  - the simple operation attr_A(p)
  - thus yields the value of attribute A
  - for that part.
- But what if, what we have is just
  - the global state σ, of the set of all monitorable parts of a given universe-of-discourse, uod,
  - the unique identifier, uid_P(p), of a part of σ, and
  - the name, ηA, of an attribute of p ?
    - Then how do we
      - ascertain the attribute value for A of p,
      - and, for *biddable* attributes A,
      - "update" p, in σ, to some A value ?
    - Here is how we express these two issues.

5.4.3.1 Evaluation of Monitorable Attributes

98. Let pi:PI be the unique identifier of any part, p, with monitorable attributes, let A be a monitorable attribute of p, and let ηA be the name of attribute A.
99. Evaluation of the [current] attribute A value of p is defined by function read_A_from_P – retr_part(pi) is defined in Sect. 5.2.5.1 on Slide 200.

value

98. pi:PI, a:A, ηA:$\eta\mathbb{T}$

99. read_A_from_P: PI × $\mathbb{T}$ → **read** σ
99. read_A(pi,ηA) ≡ attr_A(retr_part(pi))

5.4.3.2 Update of Biddable Attributes

100. The update of a monitorable attribute A, with attribute name ηA of part p, identified by pi, to a new value **write**s to the global part state σ.
101. Part p is retrieved from the global state.
102. A new part, p′ is formed such that p′ is like part p:
    (a) same unique identifier,
    (b) same mereology,
    (c) same attributes values,
    (d) except for A.
103. That new p' replaces p in σ.

```
value
98.  σ, a:A, pi:PI, ηA:ηT

100.  update_P_with_A: PI × A × ηT → write σ
100.  update_P_with_A(pi,a,ηA) ≡
101.      let p = retr_part(pi) in
102.      let p′:P ·
102a.          uid_P(p′)=pi
102b.          ∧ mereo_P(p)=mereo_P(p′)
102c.          ∧ ∀ ηA′ in analyse_attribute_type_names(p) \ {ηA}
102c.              ⇒ attr_A(p)=attr_A(p′)
102d.          ∧ attr_A(p′)=a in
103.      σ := σ \ {p} ∪ {p′}
100.      end end
```

5.4.3.3 Stationary and Mobile Attributes

- Endurants are either **stationary** or **mobile**.

Definition 51 . ***Stationary:*** An endurant is said to be stationary if it never moves ■

- Being stationary is a static attribute.

Analysis Predicate Prompt 18 . `is_stationary:`

- *The method provides the* ***domain analysis prompt****:*
  - `is_stationary` – *where* `is_stationary(e)` *holds if e is to be considered stationary* ■

Example 54 . Stationary Endurants:

- Examples of stationary endurants could be:
  - road hubs and links;
  - container terminal stacks;
  - pipeline units; and
  - sea, lake and river beds ■

Definition 52 . ***Mobile:*** An endurant is said to be mobile if it is capable of being moved – whether by its own, or otherwise ■

- Being mobile is a static attribute.

Analysis Predicate Prompt 19 . `is_mobile:`

- *The method provides the* ***domain analysis prompt****:*
  - `is_mobile` – *where* `is_mobile(e)` *holds if e is to be considered mobile* ■

Example 55 . Mobile Endurants:

- Examples of mobile endurants are:
  - automobiles;
  - container terminal vessels, containers, cranes and trucks;
  - pipeline oil (or gas, or water, ...);
  - sea, lake and river water ■
- Being stationary or mobile is an attribute of any manifest endurant.
  - Foe every manifest endurant, *e*, it is the case that
  - `is_stationary(e)≡~is_mobile(e)`.

• • •

- Being stationary or, vice-versa, being mobile
  is often **tacitly assumed.**
  - Having external or internal qualities of a certain kind
    is often also tacitly assumed.
  - A major point of the domain analysis & description approach,
    * of these lectures,
    * is to help the domain analyser cum describer –
    * the domain engineer cum researcher –
    * to unveil as many, if not all, these qualities.
  - **Tacit understanding** would not be a common problem
    was it not for us to practice it "excessively" !

5.5 SPACE and TIME

- The two concepts: space and time are not attributes of entities.
- In fact, they are not internal qualities of endurants.
- They are universal qualities of any world.
  - SPACE and TIME are unavoidable concepts of any world.
  - But we can ascribe spatial attributes to any concrete, manifest endurant.
  - And we can ascribe attributes to endurants that record temporal concepts.

5.5.1 SPACE

- Space is just there.
  - So we do not define an observer, observe_space.
  - For us – bound to model mostly artefactual worlds on this earth – there is but one space.
  - Although SPACE, as a type, could be thought of as defining more than one space we shall consider these to be isomorphic !
- SPACE is considered to consist of (an infinite number of) POINTs.

104. We can assume a point observer, observe_POINT, is a function which applies to endurants, e, and yield a point, pt : POINT

104. observe_POINT: E → POINT

  - At which "point" of an endurant, e,
    observe_$\mathbb{POINT}(e)$, is applied, or
  - which of the (infinitely) many points of an endurant E,
    observe_$\mathbb{POINT}(e)$, yields
    we leave up to the domain analyser cum describer to decide !

- We suggest, besides $\mathbb{POINT}$s, the following spatial attribute possibilities:

105. $\mathbb{EXTENT}$ as a dense set of $\mathbb{POINT}$s;
106. Volume, of concrete type, for example, m^3, as the "volume" of an $\mathbb{EXTENT}$ such that
107. $\mathbb{SURFACE}$s as dense sets of $\mathbb{POINT}$s have no volume, but an
108. Area, of concrete type, for example, m^2, as the "area" of a dense set of $\mathbb{POINT}$s;
109. $\mathbb{LINE}$ as dense set of $\mathbb{POINT}$s with no volume and no area, but
110. Length, of concrete type, for example, m.

- For these we have that

111. the *intersection*, ⋂, of two EXTENTs is an EXTENT of possibly nil Volume,
112. the intersection, ⋂, of two SURFACEs may be either a possibly nil SURFACE or a possibly nil LINE, or a combination of these.
113. the intersection, ⋂, of two LINEs may be either a possibly nil LINE or a POINT.

- Similarly we can define

114. the *union*, ⋃, of two not-disjoint EXTENTs,
115. the *union*, ⋃, of two not-disjoint SURFACEs,
116. the *union*, ⋃, and of two not-disjoint LINEs.

- and:

117. the *[in]equality*, ≠, =, of
     pairs of 𝔼𝕏𝕋𝔼ℕ𝕋,
     pairs of 𝕊𝕌ℝ𝔽𝔸ℂ𝔼s, and
     pairs of 𝕃𝕀ℕ𝔼s.

- We invite the reader to first
  - first express the signatures for these operations,
  - then their pre-conditions,
  - and finally, being courageous, appropriate fragments of axiom systems.

- We leave it up to the reader to introduce, and hence define, functions that
  - add, subtract, compare, etc.,
  - 𝔼𝕏𝕋𝔼ℕ𝕋s, 𝕊𝕌ℝ𝔽𝔸ℂ𝔼s, 𝕃𝕀ℕ𝔼s, etc.

5.5.2 Mathematical Models of Space

- Figure 5.3 on Slide 301 diagrams some mathematical models of space.
- We shall hint[14] at just one of these spaces.

[14] Figure 5.3 on Slide 301 is taken from https://en.wikipedia.org/wiki/Space_(m

5.5.2.1 Metric Spaces

Metric Space

Axiom System 1 .

- *A metric space is an ordered pair* (M,d) *where* M *is a set and* d *is a metric on* M*, i.e., a function:*

$$d : M \times M \to \textbf{\textit{Real}}$$

- *such that for any* $x,y,z \in M$*, the following holds:*

$$d(x,y) = 0 \equiv x = y \quad \textit{identity of indiscernibles} \tag{5.1}$$

$$d(x,y) = d(y,x) \quad \textit{symmetry} \tag{5.2}$$

$$d(x,z) \leq d(x,y) + d(y,z) \quad \textit{sub-additivity or triangle inequality} \tag{5.3}$$

- *Given the above three axioms, we also have that* $d(x,y) \geq 0$ *for any*

$x, y \in M$.

- *This is deduced as follows:*

$$d(x,y) + d(y,x) \geq d(x,x) \quad \textit{triangle inequality} \tag{5.4}$$

$$d(x,y) + d(y,x) \geq d(x,x) \quad \textit{by symmetry} \tag{5.5}$$

$$2d(x,y) \geq 0 \quad \textit{identity of indiscernibles} \tag{5.6}$$

$$d(x,y) \geq 0 \quad \textit{non-negativity} \tag{5.7}$$

- *The function d is also called distance function or simply distance.*
- *Often, d is omitted and one just writes M for a metric space if it is clear from the context what metric is used.*

.

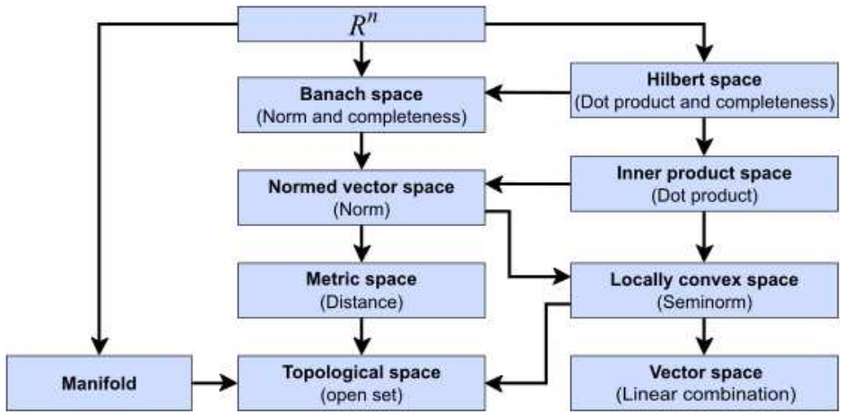

Figure 5.3: Variety of Abstract Spaces. An arrow from space A to space B implies that A is also a kind of B.

5.5.3 TIME

a moving image of eternity;
the number of the movement in respect of the before and the after;
the life of the soul in movement as it passes
from one stage of act or experience to another;
a present of things past: memory,
a present of things present: sight,
and a present of things future: expectations[15]

This thing all things devours:
Birds, beasts, trees, flowers;
Gnaws iron, bites steel,
Grinds hard stones to meal;
Slays king, ruins town,
And beats high mountain down.[16]

[15] Quoted from [1, Cambridge Dictionary of Philosophy]
[16] J.R.R. Tolkien, The Hobbit

- Concepts of time continue to fascinate philosophers and scientists [52, 29, 35, 36, 39, 40, 41, 42, 43, 44, 45] and [30].
- J.M.E. McTaggart (1908, [35, 29, 45]) discussed theories of time around the notions of
  - **"A-series":** with concepts like "past", "present" and "future", and
  - **"B-series":** has terms like "precede", "simultaneous" and "follow".
- Johan van Benthem [52] and Wayne D. Blizard [25, 1980] relates abstracted entities to spatial points and time.
- A recent computer programming-oriented treatment is given in [30, Mandrioli et al., 2013].

5.5.3.1 Time Motivated Philosophically

Definition 53 . ***Indefinite Time:***

- *We motivate the abstract notion of time as follows.*
  - *Two different states*
    must necessarily be ascribed different incompatible predicates.
    - *But how can we ensure so ?*
    - *Only if states stand in*
      an asymmetric relation to one another.
    - *This state relation is also transitive.*
    - *So that is an indispensable property of any world.*
    - *By a transcendental deduction we say that*
      primary entities exist in time.
  - *So every possible world must exist in time* ■

Definition 54 . ***Definite Time:***

- By a ***definite time*** we shall understand
  - an abstract representation of time
  - such as for example year, month, day, hour, minute, second, etc.

■

Example 56 . Temporal Notions of Endurants:

- *By temporal notions of endurants we mean*
  - *time properties of endurants,*
  - *usually modelled as attributes.*
- *Examples are:*
  - *(i) the time stamped link traffic, cf. Item 87 on Slide 263 and*
  - *(ii) the time stamped hub traffic, cf. Item 83 on Slide 259* ■

5.5.3.2 Time Values

- We shall not be concerned with any representation of time.
- That is, we leave it to the domain analyser cum describer to choose an own representation [30].
- Similarly we shall not be concerned with any representation of time intervals.[17]

118. So there is an abstract type $\mathbb{T}ime$,
119. and an abstract type $\mathbb{TI}$: $\mathbb{T}ime\mathbb{I}nterval$.
120. There is no $\mathbb{T}ime$ origin, but there is a "zero" $\mathbb{TI}$me interval.
121. One can add (subtract) a time interval to (from) a time and obtain a time.

[17] – but point out, that although a definite time interval may be referred to by number of years, number of days (less than 365), number of hours (less than 24), number of minutes (less than 60) number of seconds (less than 60), et cetera, this is not a time, but a time interval.

122. One can add and subtract two time intervals and obtain a time interval
– with subtraction respecting that
the subtrahend is smaller than or equal to the minuend.
123. One can subtract a time from another time obtaining a time interval respecting that the subtrahend is smaller than or equal to the minuend.
124. One can multiply a time interval with a real and obtain a time interval.
125. One can compare two times and two time intervals.

type
118 $\mathbb{T}$
119 $\mathbb{TI}$
value
120 **0**:$\mathbb{TI}$
121 +,−: $\mathbb{T} \times \mathbb{TI} \to \mathbb{T}$
122 +,−: $\mathbb{TI} \times \mathbb{TI} \xrightarrow{\sim} \mathbb{TI}$
123 −: $\mathbb{T} \times \mathbb{T} \to \mathbb{TI}$
124 ∗: $\mathbb{TI} \times$ **Real** $\to \mathbb{TI}$
125 <,≤,=,≠,≥,>: $\mathbb{T} \times \mathbb{T} \to$ **Bool**
125 <,≤,=,≠,≥,>: $\mathbb{TI} \times \mathbb{TI} \to$ **Bool**
axiom
121 ∀ t:$\mathbb{T}$ · t+**0** = t

5.5.3.3 Temporal Observers

126. We define the signature of the meta-physical time observer.

type
126 $\mathbb{T}$
value
126 **record**_$\mathbb{TIME}$(): **Unit** $\to$ $\mathbb{T}$

- The time recorder applies to nothing and yields a time.
- **record**_$\mathbb{TIME}$() can only occur in action, event and behavioural descriptions.

5.5.3.4 "Soft" and "Hard" Real-time

- We loosely identify a spectrum of from "soft" to "hard" temporalities — through some informally worded texts.
- On that background we can introduce the term 'real-time'.
- And hence distinguish between 'soft' and 'hard' real-time issues.
- From an example of trying to formalise these in RSL,
- we then set the course for these next lectures.

5.5.3.4.1 Soft Temporalities

- You have often wished, we assume, that
  "your salary never goes down, say between your ages of 25 to 65".
- How to express that?
- Taking into account other factors, you may additionally wish that
  "your salary goes up."
- How do we express that?
- Taking also into account that your job is a seasonal one, we may need to refine the above into
  "between un-employments your salary does not go down".
- How now to express that?

5.5.3.4.2 Hard Temporalities

- The above quoted ("...") statements may not have convinced you about the importance of speaking precisely about time, whether narrating or formalising.
- So let's try some other examples:
  - *"The alarm clock must sound exactly at 6 am unless someone has turned it off sometime between 5am and 6 am the same morning."*
  - *"The gas valve must be open for exactly 20 seconds every 60 seconds."*

  - *"The sum total of time periods — during which the gas valve is open and there is no flame consuming the gas — must not exceed one twentieth of the time the gas valve is open."*
  - *"The time between pressing an elevator call button on any floor and the arrival of the cage and the opening of the cage door at that floor must not exceed a given time* $t_{arrival}$*".*
- The next lecture items will hint at ways and means of speaking of time.

5.5.3.4.3 Soft and Hard Real-time

- The informally worded temporalities of "soft real-time"
  - can be said to involve time in a very "soft" way:
  - No explicit times (eg., 15:45:00), deadlines
    (eg., *"27'th February 2004"*),
    or time intervals
    (eg., *"within 2 hours"*),
    were expressed.
- The informally worded temporalities of "hard real-time", in contrast,
  - can be said to involve time in a "hard" way:
  - Explicit times were mentioned.

- For pragmatic reasons, we refer to the former examples, the former "invocations" of 'temporality', as being representative of soft real-time,
- whereas we say that the latter invocations are typical of hard real-time.
- Please do not confuse the issue of soft versus hard real-time:
  - It is as much hard real-time if we say that something must happen two light years and five seconds from tomorrow at noon!

Example 57 . Soft Real-Time Models Expressed in Ordinary RSL Logic:

- Let us assume a salary data base SDB
- which at any time records your salary.
- In the conventional way of modelling time in RSL we assume that SDB maps time into Salary:

```
type
  Time, Sal
  SDB = Time ~m~> Sal
value
  hi: (Sal×Sal)|(Time×Time) → Bool
  eq: (Sal×Sal)|(Time×Time) → Bool
  lo: (Sal×Sal)|(Time×Time) → Bool
axiom
  ∀ σ:SDB,t,t′:Time · {t,t′}⊆dom σ ∧ hi(t′,t) ⇒ ~lo(σ(t′),σ(t))
  ∀ t,t′:Time ·
    (hi(t′,t)≡~(eq(t′,t)∨lo(t′,t))) ∧
    (lo(t′,t)≡~(eq(t′,t)∨hi(t′,t))) ∧
    (eq(t′,t)≡~(lo(t′,t)∨hi(t′,t))) ... /* same for Sal */
```

Example 58 . Hard Real-Time Models Expressed in "Ordinary" RSL Logic:

- To express hard real-time using just RSL we must assume a demon, a process which represents the clock:

```
type
  𝕋 = Real
value
  time: Unit → 𝕋
  time() as t
axiom
  time() ≠ time()
```

- The axiom is informal:
  - It states that no two invocations of the time function yields the same value.
  - But this is not enough.
  - We need to express that “immediately consecutive” invocations of the time function yields “adjacent” time points.
- $\mathbb{T}$ provides a linear model of real-time.

```
variable
  t1,t2 : 𝕋
axiom
  □ (t1 := time();
     t2 := time();
     t2 – t1 = /∗ infinitesimally small time interval: 𝕋𝕀∗/ ∧
     t2 > t1 ∧ ∼∃ t:𝕋· t1 < t < t2 )
```

- $\mathbb{TI}$ provides a linear model of intervals of real-time.[18]
- The □ operator is here the "standard" RSL modal operator over states:
  - Let P be a predicate involving globally declared variables.
  - Then $\Box P$ asserts that P holds in any state (of these variables).
- But even this is not enough. Much more is needed ■

[18] Of course, we really do not need make a distinction between $\mathbb{T}$ and $\mathbb{TI}$, The former tries to model a real-time since time immemorial, i.e., the creation of the universe. If we always work with a time axis from "that started recently", i.e., a relative one, then we can "collapse" $\mathbb{T}$ and $\mathbb{TI}$ into just $\mathbb{T}$.

5.6 Intentional Pull

Left out of the TU Wien lectures

- In this part of the lecture
  we shall encircle the 'intention' concept
  by extensively quoting from Kai Sørlander's Philosphy
  [46, 47, 48, 49].
- *Intentionality*[19] "expresses" conceptual, abstract relations between otherwise, or seemingly unrelated entities.
- Intentional properties of a domain is not an internal quality of any (pair or group of) entities.
- They are potential, universal qualities of any world.

[19] The Oxford English Dictionary [34] characterises intentionality as follows: *"the quality of mental states (e.g. thoughts, beliefs, desires, hopes) which consists in their being directed towards some object or state of affairs"*.

5.6.1 Issues Leading Up to Intentionality

5.6.1.1 Causality of Purpose

- *"If there is to be the possibility of language and meaning*
  - *then there must exist primary entities*
  - *which are not entirely encapsulated within the physical conditions;*
  - *that they are stable and*
  - *can influence one another.*
- *This is only possible if such primary entities are*
  - *subject to a supplementary causality*
  - *directed at the future:*
  - *a causality of purpose."*

5.6.1.2 Living Species

- *"These primary entities are here called living species.*
- *What can be deduced about them ? They are*
  - *characterised by causality of purpose:*
  - *they have some form they can be developed to reach;*
  - *and which they must be causally determined to maintain;*
  - *this development and maintenance must occur in an exchange of matter with an environment.*
  - *It must be possible that living species occur in one of two forms:*
    - *one form which is characterised by development, form and exchange,*
    - *and another form which, additionally, can be characterised by the ability to purposeful movements.*
  - *The first we call plants, the second we call animals."*

5.6.1.3 Animate Entities

- *"For an animal to purposefully move around*
  - *there must be "additional conditions" for such self-movements to be in accordance with the principle of causality:*
    - *they must have sensory organs sensing among others the immediate purpose of its movement;*
    - *they must have means of motion so that it can move; and*
    - *they must have instincts, incentives and feelings as causal conditions that what it senses can drive it to movements.*
  - *And all of this in accordance with the laws of physics."*

5.6.1.4 Animals

"To possess these three kinds of "additional conditions",

- *must be built from special units which have an inner relation to their function as a whole;*
- *Their purposefulness must be built into their physical building units,*
- *that is, as we can now say, their genomes.*
- *That is, animals are built from genomes which give them the inner determination to such building blocks for instincts, incentives and feelings.*
- *Similar kinds of deduction can be carried out with respect to plants.*
- *Transcendentally one can deduce basic principles of evolution but not its details."*

5.6.1.5 Humans – Consciousness and Learning

- *"The existence of animals is a necessary condition for there being language and meaning in any world.*
  - *That there can be language means that animals are capable of developing language.*
  - *And this must presuppose that animals can learn from their experience.*
  - *To learn implies that animals*
    - *can feel pleasure and distaste*
    - *and can learn.*
  - *One can therefore deduce that animals must possess such building blocks whose inner determination is a basis for learning and consciousness."*

- *"Animals with higher social interaction*
  - *uses signs, eventually developing a language.*
  - *These languages adhere to*
    the same system of defined concepts
  - *which are a prerequisite for any description of any world:*
    - *namely the system that philosophy lays bare from a basis*
    - *of transcendental deductions and*
    - *the principle of contradiction and*
    - *its implicit meaning theory.*
- *A human is an animal which has a language."*

5.6.1.6 Knowledge

- *"Humans must be conscious*
  - *– of having knowledge of its concrete situation,*
  - *– and as such that humans*
    can have knowledge about what they feel
  - *– and eventually that humans can know whether*
    what they feel is true or false.
  - *– Consequently a human can describe his situation correctly."*

5.6.1.7 Responsibility

- *"In this way one can deduce that humans*
  - *can thus have memory*
  - *and hence can have responsibility,*
  - *be responsible.*
  - *Further deductions lead us into ethics."*

• • •

- We shall not further develop the theme of
  - *living species: plants and animals,*
  - thus excluding, most notably *humans,*
  - in this chapter.

- We claim that the present chapter,
  - due to its foundation in Kai Sørlander's Philosophy,
  - provides a firm foundation
  - within which we, or others, can further develop
  - this theme: *analysis & description of living species.*

• • •

5.6.2 Intentionality

- *Intentionality* as
  - a philosophical concept
  - is defined by the
    Stanford Encyclopedia of Philosophy[20] as
    * *"the power of minds*
      to be about,
      to represent, or
      to stand for,
    * *things, properties and states of affairs."*

[20] Jacob, P. (Aug 31, 2010). *Intentionality*. Stanford Encyclopedia of Philosophy (https://seop.illc.uva.nl/entries/intentionality/) October 15, 2014, retrieved April 3, 2018.

5.6.2.1 Intentional Pull

- Two or more artefactual parts
  - of different sorts, but with overlapping sets of intents
  - may excert an *intentional "pull"* on one another.
- This *intentional "pull"* may take many forms.
  - Let $p_x : X$ and $p_y : Y$
  - be two parts of *different sorts* (X, Y),
  - and with *common intent*, ι.
  - *Manifestations* of these, their common intent
  - must somehow be *subject to constraints*,
  - and these must be *expressed predicatively*.

Example 59 . Double Bookkeeping:

- A classical example of intentional pull
  is found in double bookkeeping
  - which states that every financial transaction
  - has equal and opposite effects in at least two different accounts.
  - It is used to satisfy the accounting equation:
    $Assets = Liabilities + Equity$.
  - The intentional pull is then reflected in commensurate postings,
    for example:
    * either in both debit and passive entries
    * or in both credit and passive entries.

- When a compound artefact
  - is modelled as put together
    with a number of distinct sort endurants
  - then it does have an intentionality and
  - the components' individual intentionalities does,
    i.e., shall relate to that.
    * The composite road transport system has intentionality
      of the road serving the automobile part, and
    * the automobiles have
      the intent of being served by the roads,
      across "a divide", and vice versa,
      the roads of serving the automobiles.

- Natural endurants, for example,
  - rivers, lakes, seas[21] and oceans become, in a way, artefacts when mankind use them for transport;
  - natural gas becomes an artefact when drilled for, exploited and piped; and
  - harbours make no sense without artefactual boats sailing on the natural water.

[21] Seas are smaller than oceans and are usually located where the land and ocean meet. Typically, seas are partially enclosed by land. The Sargasso Sea is an exception. It is defined only by ocean currents [oceanservice.noaa.gov/facts/oceanorsea.html].

5.6.2.2 The Type Intent

- This, perhaps vague, concept of intentionality
  has yet to be developed into something of a theory.
- Despite that this is yet to be done,
  we shall proceed to define an *intentionality analysis function*.
- First we postulate a set of **intent designators**.
  - An *intent designator* is really a further undefined quantity.
  - But let us, for the moment,
    think of them as simple character strings, that is, literals,
    for example `""transport"`, `"eating"`, `"entertainment"`, etc.

```
type Intent
```

5.6.2.3 Intentionalities

Observer Function Prompt 10. `analyse_intentionality:`

- *The domain analyser analyses an endurant as to the finite number of intents, zero or more, with which the analyser judges the endurant can be associated.*
- *The method provides the* **domain analysis prompt**:
  - `analyse_intentionality`

directs the domain analyser to observe a set of intents.

value analyse_intentionality(e) ≡ {i_1,i_2,...,i_n}⊆Intent

Example 60 . Intentional Pull: Road Transport:

- We simplify the link, hub and automobile histories –
- aiming at just showing an essence of the intentional pull concept.

127. With links, hubs and automobiles
we can associate history attributes:
   (a) link history attributes time-stamped records,
   as an ordered list, the presence of automobiles;
   (b) hub history attributes time-stamped records,
   as an ordered list, the presence of automobiles; and
   (c) automobile history attributes time-stamped records,
   as an ordered list, their visits to links and hubs.

type
127a. LHist = AI $\xrightarrow[m]{}$ $\mathbb{TIME}^*$
127b. HHist = AI $\xrightarrow[m]{}$ $\mathbb{TIME}^*$
127c. AHist = (LI|HI) $\xrightarrow[m]{}$ $\mathbb{TIME}^*$

value
127a. attr_LHist: L → LHist
127b. attr_HHist: H → HHist
127c. attr_AHist: A → AHist

5.6.2.4 Wellformedness of Event Histories

- Some observations must be made with respect to the above modelling of time-stamped event histories.

128. Each $\tau_\ell : \mathbb{TIME}^*$ is an indefinite list.
We have not expressed any criteria
for the recording of events: *all the time, continuously* ! (?)
129. Each list of times, $\tau_\ell : \mathbb{TIME}^*$, is here to be in decreasing, *continuous* order of times.
130. Time intervals from when an automobile enters a link (a hub)
till it first time leaves that link (hub)
must not overlap with other such time intervals for that automobile.
131. If an automobile leaves a link (a hub), at time τ, then it may enter a hub (resp. a link)
and then that must be at time τ'
where τ' is some infinitesimal, sampling time interval, quantity larger that τ.
Again we refrain here from speculating on the issue of sampling !
132. Altogether, ensembles of link and hub event histories
for any given automobile define routes that automobiles travel across the road net.
Such routes must be in the set of routes defined by the road net.

- As You can see, there is enough of interesting modelling issues to tackle !

5.6.2.5 Formulation of an Intentional Pull

133. An *intentional pull* of any road transport system, *rts*, is then if:

(a) for any automobile, a, of rts,
on a link, ℓ (hub, h),
at time τ,

(b) then that link, ℓ, (hub h)
"records" automobile a
at that time.

134. and:

(c) for any link, ℓ (hub, h)
being visited by an automobile, a,
at time τ,

(d) then that automobile, a,
is visiting that link, ℓ (hub, h),
at that time.

axiom
133a. ∀ a:A · a ∈ *as* ⇒
133a. **let** ahist = attr_AHist(a) **in**
133a. ∀ ui:(LI|HI) · ui ∈ **dom** ahist ⇒
133b. ∀ τ:$\mathbb{TIME}$ · τ ∈ **elems** ahist(ui) ⇒
133b. **let** hist = is_LI(ui) → attr_LHist(retr_L(ui))(σ),
133b. _ → attr_HHist(retr_H(ui))(σ) **in**
133b. τ ∈ **elems** hist(uid_A(a)) **end end**
134. ∧
134c. ∀ u:(L|H) · u ∈ $ls \cup hs$ ⇒
134c. **let** uhist = attr(L|H)Hist(u) **in**
134d. ∀ ai:AI · ai ∈ **dom** uhist ⇒
134d. ∀ τ:$\mathbb{TIME}$ · τ ∈ **elems** uhist(ai) ⇒
134d. **let** ahist = attr_AHist(retr_A(ai))(σ) **in**
134d. τ ∈ **elems** uhist(ai) **end end**

- Please note, that *intents* are not [thought of as] attributes.
  - We consider *intents* to be a fourth,
    a comprehensive internal quality of endurants.
  - They, so to speak, govern relations between the three other
    internal quality of endurants:
    the unique identifiers, the mereologies and the attributes.
  - That is, they predicate them, "arrange" their comprehensiveness.
- Much more should be said about intentionality.
- It is a truly, I believe, worthy research topic of its own ■

Example 61 . Aspects of Comprehensiveness of Internal Qualities:

- Let us illustrate the issues "at play" here.
  - Consider a road transport system uod.
    - Applying `analyse_intentionality`(uod) may yield the set {`"transport", ...`}.
  - Consider a financial service industry, fss.
    - Applying `analyse_intentionality`(fss) may yield the set {`"interest on deposit", ...`}.
  - Consider a health care system, hcs.
    - Applying `analyse_intentionality`(hcs) may yield the set {`"cure diseases", ...`}.
- What these analyses of intentionality yields, with respect to expressing intentional pull, is entirely of the discretion of the *domain analyser & describer* ■

- We bring the above example,
  Example 61 on the previous slide, to indicate,
  as the name of the example reveals,
  "Aspects of Comprehensiveness of Internal Qualities".
  - That the various components of artefactual systems relate in – further to be explored – ways.
  - In this respect, performing domain analysis & description is not only an engineering pursuit, but also one of research.
  - We leave it to the students to pursue this research aspect of domain analysis & description.

5.6.3 Artefacts

- Humans create artefacts –
  for a reason, to serve a purpose, that is, with intent.
  - Artefacts are like parts.
  - They satisfy the laws of physics –
  - and serve a *purpose*, fulfill an *intent*.

5.6.4 Assignment of Attributes

- So what can we deduce from the above, almost three pages ?
- The attributes of natural parts and natural fluids
  - are generally of such concrete types –
  - expressible as some **real** with a dimension[22] of
  - the International System of Units:
  - `https://physics.nist.gov/cuu/Units/units.html`.
- Attribute values usually enter into *differential equations* and *integrals*,
- that is, classical calculus.

[22] Basic units are *m*eter, *k*ilogram, *s*econd, *A*mpere, *K*elvin, *mol*e, and *c*an*d*ela. Some derived units are: *N*ewton: $kg{\times}m{\times}s^{-2}$, *W*e*b*er: $kg \times m^2 \times s^{-2} \times A^{-1}$, etc.

- The attributes of humans, besides those of parts,
  - significantly includes one of a
    usually non-empty set of *intents*.
    * In directing the creation of artefacts
    * humans create these with an intent.

Example 62 . Intentional Pull: General Transport:

- These are examples of human intents:
  - they create *roads* and *automobiles*
    with the intent of *transport*,
  - they create *houses*
    with the intents of *living, offices, production*, etc., and
  - they create *pipelines*
    with the intent of *oil* or *gas transport* ■
- Human attribute values usually enter into *modal logic* expressions.

5.6.5 Galois Connections

- Galois Theory was first developed by Évariste Galois [1811-1832] around 1830[23].
- Galois theory emphasizes a notion of **Galois connections**.
- We refer to standard textbooks on Galois Theory, e.g., [51, 2009].

[23] en.wikipedia.org/wiki/Galois_theory

5.6.5.1 Galois Theory: An Ultra-brief Characterisation

- To us, an essence of Galois connections can be illustrated as follows:
  - Let us observe[24] properties of a number of endurants,
    say in the form of attribute types.
  - Let the function $\mathcal{F}$ map sets of entities to the set of common attributes.
  - Let the function $\mathcal{G}$ map sets of attributes
    to sets of entities that all have these attributes.
  - $(\mathcal{F},\mathcal{G})$ is a Galois connection
    - if, when including more entities,
      the common attributes remain the same or fewer, and
    - if when including more attributes,
      the set of entities remain the same or fewer.
    - $(\mathcal{F},\mathcal{G})$ is monotonously decreasing.

[24] The following is an edited version of an explanation kindly provided by Asger Eir, e-mail, June 5, 2020 [27, 28, 21].

Example 63 . LEGO Blocks:

- We[25] have
  - There is a collection of LEGO™ blocks.
  - From this collection, A, we identify the red square blocks, e.
  - That is $\mathcal{F}(A)$ is B = {attr_Color(e) = red,attr_Form(e)=**square**}.
  - We now add all the blue square blocks.
  - And obtain A'.
  - Now the common properties are their **squareness**:
    $\mathcal{F}(A')$ is B' = {attr_Form(e)=**square**}.
  - More blocks as argument to $\mathcal{F}$ yields fewer or the same number of properties.
  - The more entities we observe, the fewer common attributes they possess ■

[25] The E-mail, June 5, 2020, from Asger Eir

Example 64 . Civil Engineering: Consultants and Contractors:

Less playful, perhaps more seriously, and certainly more relevant to our endeavour, is this next example.

- Let X be the set of civil engineering, i.e., building, consultants, i.e., those who, like architects and structural engineers design buildings – of whatever kind.
- Let Y be the set of building contractors, i.e., those firms who actually implement, i.e., build to, those designs.
- Now a subset, $X_{bridges}$ of X, contain exactly those consultants who specialise in the design of bridges,
  with a subset, $Y_{bridges}$, of Y capable of building bridges.
- If we change to a subset, $X_{bridges,tunnels}$ of X, allowing the design of both bridges **and** tunnels, then we obtain a corresponding subset, $Y_{bridges,tunnels}$, of Y.

- So when
  - we enlarge the number of properties from ‘bridges’ to ‘bridges and tunnels’,
  - we reduce, most likely, the number of contractors able to fulfill such properties,
  - and vice versa,
- then we have a Galois Connection[26] ■

[26] This was, more formally, shown Dr. Asger Eir’s PhD thesis [27].

5.6.5.2 Galois Connections and Intentionality – A Possible Research Topic ?

- We have a hunch[27] !
  - Namely that there are some sort of Galois Connections with respect to intentionality.
- We leave to the interested student to pursue this line of inquiry.

[27] Hunch: a feeling or guess based on intuition rather than fact.

5.6.6 Discovering Intentional Pulls

- The analysis and description of a domain's
  - external qualities and
  - the internal qualities of
    unique identifiers, mereologies and attributes
  - can be pursued systematically –
  - endurant sort by sort.

- Not so with the discovery of
  a domain's possible intentional pulls.
- Basically *"what is going on"* here is
  - that the domain analyser cum describer
  - considers pairs, triples or more
    part "independent"[28] endurants
  - and reflects on whether they stand
    in an *intentional pull*
    relation to one another.
- We refer to Sects. 5.6.2.2 – 5.6.2.3.

[28] By "independent" we shall here mean that these endurants are not 'derived' from one-another !

5.7 A Domain Discovery Procedure, II

- We continue from Sect. 4.8.

5.7.1 The Process

- We shall again emphasize some aspects of the *domain analyser & describer* method.
  - A **method procedures** is that of *exhaustively analyse & describe* all internal qualities of the domain under scrutiny.
  - A **method technique** implied here is that sketched below.
  - The **method tools** are here all the analysis and description prompts covered so far.

- Please be reminded of *Discovery Schema 0*'s declaration of *Notice Board* variables (Slide 156).
- In this section of the lecture we collect
  - the *description of unique identifiers* of all parts of the state;
  - the *description of mereologies* of all parts of the state; and
  - the *description of attributes* of all parts of the state.
- We finally gather these into the *discover_internal_endurant_qualities* procedures.

An Endurant Internal Qualities Domain Analysis and Description Process, I

```
value
  discover_uids: Unit → Unit
  discover_uids() ≡
    for ∀ v • v ∈ gen
      do txt := txt † [ type_name(v)↦txt(type_name(v))⌢⟨describe_unique_identifier(v)⟩ ] end
  discover_mereologies: Unit → Unit
  discover_mereologies() ≡
    for ∀ v • v ∈ gen
      do txt := txt † [ type_name(v)↦txt(type_name(v))⌢⟨describe_mereology(v)⟩ ] end
  discover_attributes: Unit → Unit
  discover_attributes() ≡
    for ∀ v • v ∈ gen
      do txt := txt † [ type_name(v)↦txt(type_name(v))⌢⟨describe_attributes(v)⟩ ] end
discover_intentional_pulls: Unit → Unit
discover_intentional_pulls() ≡
    for ∀ (v′,v″) • {v′,v″} ⊆ gen
      do txt := txt † [ type_name(v′)↦txt(type_name(v′))⌢⟨describe_intentional_pull()⟩ ]
            † [ type_name(v″)↦txt(type_name(v″))⌢⟨describe_intentional_pull()⟩ ] end
describe_intentional_pull: Unit → ...
describe_intentional_pull() ≡ ...
```

An Endurant Internal Qualities Domain Analysis and Description Process, II

```
value
  discover_internal_qualities: Unit → Unit
  discover_internal_qualities() ≡
    discover_uids() ;
      axiom [ all parts have unique identifiers ]
    discover_mereologies() ;
      axiom [ all unique identifiers are mentioned in sum total of ]
        [ all mereologies and no isolated proper sets of parts ]
    discover_attributes() ;
      axiom [ sum total of all attributes span all parts of the state ]
    discover_intentional_pulls()
```

- We shall comment on the axioms in the next section.

5.7.2 A Suggested Analysis & Description Approach, II

- Figure 4.3 on Slide 127 possibly hints at an analysis & description order in which
  - not only the external qualities of endurants are analysed & described,
  - but also their internal qualities of unique identifiers, mereologies and attributes.
- In Sect. 4.8 on Slide 154 we were concerned with the analysis & description order of endurants.

- We now follow up on the issue of (in Sect. 4.5.1.3 on Slide 125) on how compounds are treated: namely as both a "root" parts and as a composite of two or more "sibling" parts and/or fluids.
  - The taxonomy of the road transport system domain, cf. Fig. 4.3 on Slide 127 and Example 29 on Slide 104, thus gives rise to many different analysis & description traversals.
  - Figure 5.4 on the next slide illustrates one such order.

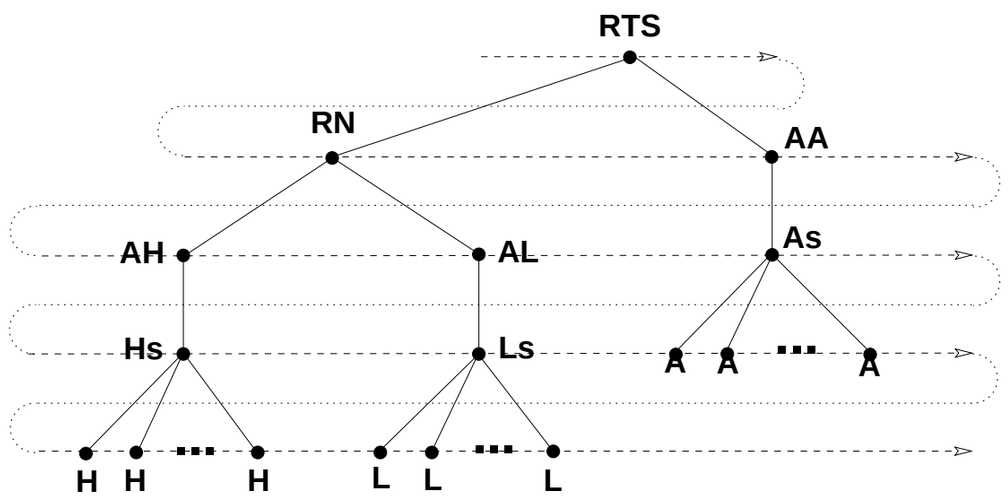

Figure 5.4: A Breadth-First, Top-Down Traversal

– Again, it is up to the domain engineer cum scientist to decide.
  * If the domain analyser cum describer decides to not endow a compound "root" with internal qualities,
  * then an 'internal qualities' traversal will not have to neither analyse nor describe those qualities.

5.8 Summary

#	Name	Introduced
	Analysis Predicates	
16	is_manifest	page 175
17	is_structure	page 175
	Attribute Analysis Predicates	
1	is_static_attribute	page 241
2	is_dynamic_attribute	page 243
3	is_inert_attribute	page 244
4	is_reactive_attribute	page 246
5	is_active_attribute	page 248
6	is_autonomous_attribute	page 249
7	is_biddable_attribute	page 251
8	is_programmable_attribute	page 253
9	is_monitorable_only_attribute	page 257
	Analysis Functions	
	all_uniq_ids	page 192
	calculate_all_unique_identifiers	page 191
6	analyse_attribute_types	page 271
7	sta_attr_types	page 272
8	mon_attr_types	page 273
9	pro_attr_types	page 274
	Retrieval, Read and Write Functions	
	retr_part	page 200
99	read_A_from_P	page 285
100	update_P_with_A	page 286
	Description Functions	
5	describe_unique_identifier	page 185
6	describe_mereology	page 209
7	describe_attributes	page 235
	Domain Discovery	
	discover_uids	page 362
	discover_mereologies	page 362
	discover_attributes	page 362
	discover_internal_qualities	page 362

• • •

- Please consider Fig. 4.1 on Slide 63.
  - This chapter has covered the horisontal and vertical lines to the left in Fig. 4.1.