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.
<table>
<thead>
<tr>
<th>Lecture # 1</th>
<th>Monday 24 October 2022</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Domain Overview</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture # 2</th>
<th>Tuesday 25 October 2022</th>
<th>Endurants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>External Qualities, Analysis</td>
<td>47–126</td>
</tr>
<tr>
<td></td>
<td>External Qualities, Synthesis</td>
<td>133–162</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture # 3</th>
<th>Thursday 27 October 2022</th>
<th>Endurants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Internal Qualities, Unique Identifiers</td>
<td>163–199</td>
</tr>
<tr>
<td></td>
<td>Internal Qualities, Mereology</td>
<td>200–226</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture # 4</th>
<th>Friday 28 October 2022</th>
<th>Endurants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Internal Qualities, Attributes</td>
<td>228–329</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture # 5</th>
<th>Monday 31 October 2022</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example</td>
<td>402–483</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture # 6</th>
<th>Thursday 3 November 2022</th>
<th>Perdurants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The “Discrete Statics”</td>
<td>330–362</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture # 7</th>
<th>Friday 4 November 2022</th>
<th>Perdurants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The “Discrete Dynamics”</td>
<td>363–401</td>
</tr>
<tr>
<td></td>
<td>Summary Discussion</td>
<td></td>
</tr>
</tbody>
</table>
Preface

• These lectures expound a method:

• By a **method** we shall understand
  – a set of **principles** and **procedures**
  – for selecting and applying a set of
  – **techniques** and **tools**
  – in order to achieve an orderly construction
  – of a **solution** to a **problem**.

• By **methodology** we shall understand
  – the **study & application** of one or more methods.
• By a **formal method** we shall understand a method
  – whose decisive **principles** include that of
    considering its artefacts as *mathematical* quantities;
  – whose decisive **procedures** include those of
  – whose decisive **techniques** include those of
  – whose decisive **tools** include those of
    one or more **formal languages**
• By a language we shall here understand a set of strings of characters, i.e., sentences,
  – sentences which are structured according to some syntax, i.e., grammar,
  – are given meaning by some semantics,
  – and are used according to some pragmatics.

• By a formal language we shall here understand a language
  – whose syntax and semantics can both be expressed mathematically
  – and for whose sentences one can rationally reason (argue, prove) properties.

• In these lectures we shall especially enunciate these:
  – principles,
  – procedures,
  – techniques, and
  – tools.
Lecture 1: Domains

• In this lecture, i.e., the next 45 mins.,
  – I shall survey “all” of the most important
  – aspect of Domain Analysis & Description.
• These will all be further explained
  and more aspects will be introduced
  in all the subsequent lectures.
CHAPTER 3. Domains

3.1 Domain Definition

Definition 1. *Domain:* By a *domain* we shall understand

• a *rationally describable* segment of

• a *discrete dynamics* segment of

• a *human assisted reality*, i.e., of the world;

• its *solid or fluid entities*:
  – *natural* [“God-given”] and
  – *artefactual* [“man-made”],

• and its *living species entities*:

• *plants* and *animals* – including, notably, *humans*
Example 1. **Domains:** A few, more-or-less self-explanatory examples:

- **Rivers** – with their natural sources, deltas, tributaries, waterfalls, etc., and their man-made dams, harbours, locks, etc. [25]
- **Road nets** – with street segments and intersections, traffic lights, and automobiles.
- **Pipelines** – with their wells, pipes, valves, pumps, forks, joins and wells [14].
- **Container terminals** – with their container vessels, containers, cranes, trucks, etc. [20]
• The definition relies on the understanding of the terms ‘rationally describable’, ‘discrete dynamics’, ‘human assisted’, ‘solid’ and ‘fluid’.

• The last two will be explained later.

• By rationally describable we mean that what is described can be understood, including reasoned about, in a rational, that is, logical manner.

• By discrete dynamics we imply that we shall basically, in this primer, rule out such domain phenomena which have properties which are continuous with respect to their time-wise, i.e., dynamic, behaviour.

• By human-assisted we mean that the domains – that we are interested in modelling – have, as an important property, that they possess man-made entities.
3.2 Phenomena and Entities

- **Definition 2. Phenomena:** By a *phenomenon* we shall understand a fact that is observed to exist or happen.
  - Some phenomena are rationally describable – to a large or full degree – others are not.

- **Definition 3. Entities:** By an *entity* we shall understand a more-or-less rationally describable phenomenon.

- **Example 2. Phenomena and Entities:** Some, but not necessarily all aspects of a river can be rationally described, hence can be still be considered entities. Similarly, many aspects of a road net can be rationally described, hence will be considered entities.
3.3 Endurants and Perdurants

3.3.1 Endurants

- **Definition 4. Endurants:** those quantities of domains that we can observe (see and touch), in *space*, as “complete” entities at no matter which point in *time* – “material” entities that persists, endures

**Example 3. Endurants:** a street segment [link], a street intersection [hub], an automobile

- Domain endurants, when eventually modelled in software, typically become data. Hence the careful analysis of domain endurants is a prerequisite for subsequent careful conception and analyses of data structures for software, including data bases.
3.3.2 Perdurants

• **Definition 5.** *Perdurants* those quantities of domains for which only a fragment exists, in *space*, if we look at or touch them at any given snapshot in *time*.

**Example 4.** Perdurant: a moving automobile.

• Domain perdurants, when eventually modelled in software, typically become processes. Hence the careful analysis of domain perdurants is a prerequisite for subsequent careful conception and analyses of functions (procedures).
3.4 External and Internal Endurant Qualities

3.4.1 External Qualities

Definition 6. *External qualities:* of endurants of a manifest domain

- are, in a simplifying sense, those we can
  - sea,
  - touch and
  - have spatial extent.

- They, so to speak, take form.
Example 5. **External Qualities:**

- The Cartesian\(^1\)
  - of sets of solid atomic street intersections, and
  - of sets of solid atomic street segments, and
  - of sets of solid automobiles
  of a road transport system

- where the

  - Cartesian,  
  - sets,  
  - atomic, and  
  - solid

reflect external qualities  

---

\(^1\) Cartesian after the French philosopher, mathematician, scientist René de Descartes (1596–1650)
3.4.1.1 Discrete or Solid Endurants

Definition 7. Discrete or Solid Endurants: By a solid [or discrete] endurant we shall understand an endurant

- which is separate, individual or distinct in form or concept,
- or, rephrasing: have ‘body’ [or magnitude] of three-dimensions: length, breadth and depth [51, Vol. II, pg. 2046]

Example 6. Solid Endurants:

- The
  - wells,  - valves,  - forks,  - sinks
  - pipes,  - pumps,  - joins and

  of pipelines are solids.
- [These units may, however, and usually will, contain fluids, e.g., oil, gas or water]
• We shall mostly be analysing and describing solid endurants.
• As we shall see, in the next section,
  – we analyse and describe solid endurants as
  – either parts
  – or living species: animals and humans.
• We shall mostly be concerned with parts.
  – That is, we shall just, as: “in passing”,
  – for sake of completeness,
  – mention living species!
3.4.1.2 Fluids

• Definition 8. Fluid Endurants:
  – By a fluid endurant we shall understand an endurant which is
    * prolonged, without interruption,
      in an unbroken series or pattern;
    * or, rephrasing: a substance (liquid, gas or plasma) having the property of flowing, consisting of particles that move among themselves [51, Vol. I, pg. 774] ■

Example 7. Fluid Endurants:

• water,   • gas,  • smoke ■
• oil,     • compressed air,
• Fluids are otherwise
  – liquid, or
  – gaseous, or
  – plasmatic, or
  – granular\(^2\), or
  – plant products, i.e., chopped sugar cane, threshed, or otherwise\(^3\),
  – et cetera.

• Fluid endurants will be analysed and described in relation to solid endurants, viz. their “containers”.

\(^2\) This is a purely pragmatic decision. “Of course” sand, gravel, soil, etc., are not fluids, but for our modelling purposes it is convenient to “compartmentalise” them as fluids!

\(^3\) See footnote 2.
3.4.1.3 Parts

- Definition 9. *Parts:* The non-living species solids are what we shall call parts.
- Parts are the “work-horses” of man-made domains.
- That is, we shall mostly be concerned with the analysis and description of endurants into parts.

**Example 8. Parts:** The previous example of solids was also an example of parts.
- We distinguish between atomic and compound parts.
3.4.1.3.1 Atomic Parts

Definition 10. Atomic Part, I:

• By an atomic part we shall understand a part
  – which the domain analyser considers to be indivisible
  – in the sense of not meaningfully,
  – for the purposes of the domain under consideration,
  – that is, to not meaningfully consist of sub-parts

© Dines Bjørner, September 8, 2022
The TU Wien Lectures, Fall 2022
3.4.1.3.2 Compound Parts

- We, pragmatically, distinguish between
  - Cartesian-product-, and
  - set-oriented parts.

- If Cartesian-oriented, to consist of two or more distinctly sort-named endurants (solids or fluids),

- If set-oriented, to consist of an indefinite number of zero, one or more parts.

Definition 11. **Compound Part, I:**

- **Compound part**s are those which are
  - either Cartesian-product-
  - or are set-

- oriented parts
Example 9. **Compound Parts:** A road net consisting of

- a set of hubs, i.e., street intersections or “end-of-streets”, and
- a set of links, i.e., street segments (with no contained hubs),

is a Cartesian compound;

- and the sets of hubs and the sets of links

are part set compounds.
3.4.2 An Aside: An Upper Ontology

• We have been reasonably careful
  – to just introduce and state informal definitions
  – of phenomena and some classes thereof.

• In the next chapter we shall, in a sense, “repeat” coverage of these phenomena.
  – But now in a more analytic manner.
  – Figure 3.1 on the next slide is intended to indicate this.
Figure 3.1: Upper Ontology
• So far we have only touched upon
  – the ‘External Qualities’ labeled, dotted-dashed box
  – of the ‘Endurants’ label-led dashed box of Fig. 3.1.
• In Chapter 4 we shall treat external qualities in more depth —
• more systematically: analytically and descriptively.
3.4.3 Internal quality

Definition 12. Internal qualities:
• those properties [of endurants]
• that do not occupy space
• but can be measured or spoken about

Example 10. Internal qualities:
• the unique identity of a part,
• the relation of part to other parts, and
• the endurant attributes such as temperature, length, colour
3.4.3.1 Unique identity

• **Definition 13.** *Unique identity:*
  – an immaterial property
  – that distinguishes two *spatially* distinct solids

**Example 11.** *Unique identities:*

• Each hub in a road net is unique identified,
• so is each link
• and automobile
3.4.3.2 Mereology

• **Definition 14. Mereology:** a theory of [endurant] part-hood relations:
  – of the relations of an [endurant] parts to a whole
  – and the relations of [endurant] parts to [endurant] parts within that whole

**Example 12. Mereology:**

• that a link is topologically *connected* to exactly two specific hubs,
• that hubs are *connected* to zero, one or more specific links,
• and that links and hubs are *open* to specific subsets of automobiles
3.4.3.3 Attribute

Definition 15. Attributes: Properties of endurants

- that are not spatially observable,
- but can be either physically
- (electronically, chemically, or otherwise)
- measured or can be objectively spoken about

Example 13. Attribute: Links have

- lengths, and,
- at any one time,
- zero, one or more automobiles are occupying the links
3.5 Prompts

3.5.1 Analysis Prompts

• Definition 16. *Analysis prompt:*
  – a predicate or a function
  – that may be posed by humans to facets of a domain.
  – Observing the domain the analyser may then
  – act upon the combination of the particular prompt
  – (whether a predicate or a function,
    and then what particular one of these it is)
  – thus “applying” it to a domain phenomena,
  – and yielding, in the minds of the humans,
  – either a truth value or some other form of value
3.5.1.1 Analysis Predicate

• **Definition 17.** *Analysis predicates:*
  – an analysis prompt
  – which yields a truth value

**Example 14.** *Analysis Predicates:* General examples are
  • can an observable phenomena be rationally described, i.e., an entity,
  • is an entity a solid or a fluid.
  • is a solid endurant a part or a living species
3.5.1.2 Analysis Function

Definition 18. Analysis function:

• an analysis prompt which yields some RSL-Text

Example 15. Analysis Functions: Two examples:

• one yields the endurants of a Cartesian part and their respective sort names,

• another yields the set of a parts of a part set and their common type
3.5.2 **Description Prompt**

- **Definition 19.** *Description prompt:*
  - a function that may be posed by humans
  - who may then act upon it:
    - “applying” it to a domain phenomena, and
    - “yielding” narrative and formal **RSL-Texts**
      describing what is being observed

**Example 16.** *Description Prompts:*

- result in **RSL-Texts** describing for example a
  - (i) Cartesian endurant, or
  - (ii) its unique identifier,
  - (iii) or its mereology, or
  - (iv) its attributes,
  - (iv) or other
3.6 Perdurant Concepts

3.6.1 “Morphing” Parts into Behaviours

• As already indicated we shall
  – transcendentally deduce
  – (perdurant) behaviours from
  – those (endurant) parts
    ∗ which we, as domain analysers cum describers,
    ∗ have endowed with all three kinds of internal qualities:
      ∗ unique identifiers, mereologies and attributes.

• Chapter 6, will show how.
3.6.2 State

Definition 20. State:

• A state is any set of the parts of a domain.

Example 17. A Road System State: The domain analyser cum describer may, in brief, decide that a road system state consists of:

• the road net aggregate (of hubs and links),
• all the hubs, and all the links, and
• the automobile aggregate (of all the automobiles), and
• all the individual automobiles.

\footnote{The road net aggregate, in its perdurant form, may “model” the Department of Roads of some country, province, or town.}

\footnote{The automobile aggregate aggregate, in its perdurant form, may “model” the Department of Vehicles of some country, province, or town.}
3.6.3  **Actors**

**Definition 21.** *Actors:*

- An actor is anything that can initiate an action, an event or a behaviour.

3.6.3.1  **Action**

**Definition 22.** *Actions:*

- An action is a function that can purposefully change a state.

**Example 18.** *Road Net Actions:* These are some road net actions:

- The insertion of a new or removal of an existing hub; or
- the insertion of a new, or removal of an existing link;
3.6.3.2 Event

Definition 23. *Events:*
- An event is a function that surreptitiously changes a state.

Example 19. *Road Net Events:* These are some road net events:
- The blocking of a link due to a mud slide;
- the failing of a hub traffic signal due to power outage;
- the blocking of a link due to an automobile accident.
3.6.3.3 Behaviour

Definition 24. *Behaviours*

- a behaviour is a set of sequences of actions, events and behaviours

Example 20. *Road Net Traffic:*

- Road net traffic can be seen as a behaviour
  - of all the behaviours of automobiles,
    - where each automobile behaviour is seen as sequence of start, stop, turn right, turn left, etc., actions;
– of all the behaviours of links
  * where each link behaviour is seen
    as a set of sequences (i.e., behaviours) of “following” the
    · link entering,
    · link leaving, and
    · movement
    of automobiles on the link;
– of all the behaviours of hubs (etc.);
– of the behaviour of the aggregate of roads,
  viz. *The Department of Roads*, and
– of the behaviour of the aggregate of automobiles,
  viz, *The Department of Vehicles.*
3.6.4 Channel

- Definition 25. *Channel:*
  - A channel is anything
  - that allows synchronisation and communication
  - of values
  - between two behaviours
• We shall use Tony Hoare’s CSP concept [44]
  – to express synchronisation and communication of values between behaviours.
  – Hence the behaviour $i$ statement
    $$ch[index] ! value$$
    to state that behaviour $i$
    offers, “outputs”: $!, value$
    to behaviours indicated by $index$.
  – And behaviour $j$ expresses
    $$ch[index] ?$$
    that it is willing to accept
    “input from & synchronise with” behaviour $i, ?,$
    any $value$. 
3.7  Domain Analysis & Description

3.7.1  Domain Analysis

Definition 26.  *Domain Analysis*
• is the act of studying a domain
• as well as the result of that study
• in the form of *informal* statements

3.7.2  Domain Description

Definition 27.  *Domain Description*
• is the act of describing a domain
• as well as the result of that act
• in the form of *narratives* and *formal RSL-Text*
3.8 Closing

• This lecture has introduced the main concepts of domains such as we shall treat (analyse and describe) domains.\(^6\)

• The next lectures shall now systematically treat the analysis and description of domains.

  – That treatment takes concept by concept and
    * provides proper definitions and
    * introduces appropriate analysis and description prompts;
    * one-by-one, in an almost pedantic,
    * hence perhaps “slow” progression!

\(^6\)We have omitted treatment of living species: plants and animals – the latter including humans. They will be treated in the next chapter!
• The student may be excused
  – if they, now-and-then, loose sight of “their way”.
• Hence the present chapter.
  – To show “the way”:
  – that, for example,
    when we treat external endurant qualities,
  – there is still the internal endurant qualities,
  – and that the whole thing leads of to perdurants:
    ♦ actors,
    ♦ actions,
    ♦ events and
    ♦ behaviours.
• The next 45 minute lecture shall present fragments of a road transport system example.
Lecture 2: External Qualities, Analysis \( \frac{1}{2} \)

• This is the first, properly systematic treatment lecture of some of the
  – principles,
  – procedures,
  – techniques and
  – tools
  of the *Domain Engineering* method.
• In this lecture we cover those of
  – analysing endurants.
CHAPTER 4. Endurants: External Domain Qualities

• This, the present chapter
  – is based on Chapter 4 of [24].
  – You may wish to study that chapter for more detail.
4.1 Universe of Discourse

Definition 28. *Universe of Discourse, UoD:*

• By a *universe of discourse* we shall understand
  – the same as the *domain of interest*,
  – that is, the *domain* to be *analysed & described*
4.1.1 Identification

- The first task of a domain analyser cum describer
  - is to settle upon the domain to be analysed and described.
  - That domain has first to be given a *name*. 

4.1.2 **Naming**

• A **first decision** is to give a name to the overall domain sort,  
  * that is, the type of the domain seen as an endurant,  
  * with that sort, or type, name  
    being freely chosen by the analyser cum describer –  
  * with no such sort names having been chosen so far!
4.1.3 Examples

- Examples of UoDs
  - railways [8, 28, 10],
  - “The Market” [9],
  - container shipping [11],
  - Web systems [12],
  - stock exchange [13],
  - oil pipelines [14],
  - credit card systems [16],
  - weather information [17],
  - swarms of drones [18],
  - document systems [19],
  - container terminals [20],
  - retail systems [22],
  - assembly plants [23],
  - waterway systems [25],
  - shipping [26],
  - urban planning [29].
4.1.4 Sketching

• The **second task** of a domain analyser cum describer is to develop a *rough sketch narrative* of the domain.

• The rough-sketching of [what] a domain [is,] is not a trivial matter.
  – It is not done by a committee!
  – It usually requires repeated “trial sketches”.
  – To carry it out, i.e., the sketching, normally requires a combination of
    * physical visits to domain examples, if possible;
    * talking with domain professionals, at all levels; and
    * reading relevant literature.
  * It also includes searching the Internet for information.

• We shall show an example next.
Example 21. Sketch of a Road Transport System UoD:

• The road transport system that we have in mind consists of
  – a road net and
  – a set of automobiles (private, trucks, buses, etc.)
  – such that the road net serves to convey automobiles.

• We consider the road net to consist of
  – hubs, i.e., street intersections, including street segment connection points, and
  – links, i.e., street segments between adjacent hubs.¹

¹This “rough” narrative fails to narrate...
4.1.5 **Universe of Discourse Description**

The general universe of discourse, i.e., domain, description prompt can be expressed as follows:

**Domain Description Prompt 1.** *calc_Uiverse_of_Discourse:*

\[calc_Uiverse_of_Discourse\text{ desriber}\]

```
“Naming:
  type UoD
Rough Sketch:
  Text ”
```

The above “RSL-Text” expresses that the *calc_Uiverse_of_Discourse()* domain describer generates RSL-Text.
Here is another example rough sketch:

**Example 22. A Rough Sketch Domain Description:**

- The example is that of the production of rum, say of a **Rum Production** domain.
- From

1. the sowing, watering, and tending to of sugar cane plants;
2. via the “burning” of these prior to harvest;
3. the harvest;
4. the collection of harvest from sugar cane fields to
5. the chopping, crushing, (and sometimes repeated) boiling, cooling and centrifuging of sugar cane when making sugar and molasses (into A, B, and low grade batches);
6. the fermentation, with water and yeast, producing a ‘wash’;
7. the (pot still or column still) distilling of the wash into rum;
8. the aging of rum in oak barrels;
9. the charcoal filtration of rum;
10. the blending of rum;
11. the bottling of rum;
12. the preparation of cases of rum for sales/export; and
13. the transportation away from the rum distiller of the rum

Some comments on this example:

• Each of the enumerated items above is phrased in terms of perdurants.
  – Behind each such perdurant lies some endurant.
  – That is, in English, “every noun can be verbed”, and vice-versa.
  – So we anticipate the transcendence, from endurants to perdurants.
Method Principle 1. *From the "Overall" to The Details:*

- Our first principle, as the first task in any new domain modelling project, is
  - to “focus” on the “overall”,
  - that is, on the “entire”,
  - though specific domain
4.2 Entities

A core concept of domain modelling is that of an entity.

Definition 29. Entity:

- By an entity we shall understand a phenomenon, i.e., something
  - that can be observed, i.e., be
    * seen or touched by humans,
    * or that can be conceived
    * as an abstraction of an entity;
  - alternatively,
    * a phenomenon is an entity, if it exists, it is “being”,
    * it is that which makes a “thing” what it is: essence, essential nature.

- If a phenomenon cannot be so observed and described then it is not an entity
Analysis Predicate Prompt 1. *is_entity*:

• The domain analyser analyses “things” ($\theta$) into either entities or non-entities.

• The method provides the **domain analysis prompt**:
  – *is_entity* – where *is_entity*($\theta$) holds if $\theta$ is an entity

• *is_entity* is said to be
  – a *prerequisite prompt*
  – for all other prompts.

• *is_entity* is a method tool.

---

*marks the end of an analysis prompt definition.*
On Analysis Prompts: The \textit{is\textunderscore entity} predicate function represents the first of a number of analysis prompts.

- They are “applied” by the domain analyser to phenomena of domains.
- They yield truth values, true or false, “left” in the mind of the domain analyser.
• We have just shown how the `is_entity` predicate prompt can be applied to a universe of discourse.

• From now on we shall see prompts being applicable to successively more analysed entities.

• Figure 4.1 [Page 63] diagrams a domain description ontology of entities.

• That ontology indicates the sub-classes of endurants for which we shall motivate and for which we shall introduce
  – prompts,
  – predicates and
  – functions.
Figure 4.1: The Upper Ontology
• The present chapter shall focus only
  – on the external qualities,
  – that is, on the “contents” of the leftmost dotted box.
Method Principle 2. Justifying Analysis along Philosophical Lines:

• The concept of entities as a main focal point
  – is justified in Kai Sørlander’s philosophy.
  – Entities are in that philosophy referred to as primary objects.
  – They are the ones about which we express predicates.
4.3  Endurants and Perdurants

Method Principle 3. *Separation of Endurants and Perdurants:*

- As we shall see in this primer, the domain analysis & description method calls for the separation of
  - first considering
    * the careful analysis & description
    * of endurants,
  - then considering
    * perdurants.
- This principle is based on
  - the transcendental deduction
  - of the latter from the former
4.3.1 Endurants

Definition 30. Endurant:

• By an endurant, to repeat, we shall understand an entity
  – that can be observed, or conceived and described,
    as a “complete thing”
  at no matter which given snapshot of time;
  – alternatively an entity is endurant
    if it is capable of enduring, that is persist, “hold out”.

Were we to “freeze” time

– we would still be able to observe the entire endurant

© Dines Bjørner, September 8, 2022
Example 23. **Natural and Artefactual Endurants:**

**Geography Endurants:**

- fields,
- meadows,
- lakes,
- rivers,
- forests,
- hills,
- mountains,
- et cetera.

**Railway Track Endurants:**

- a railway track,
- its net,
- its individual tracks,
- switch points,
- trains,
- their individual locomotives,
- signals,
- et cetera.
Road Transport System Endurants:

• the transport system,
• its road net aggregate and the aggregate of automobiles,
• the set of links (road segments) and hubs (road intersections) of the road net aggregate,
• these links and hubs, and
• the automobiles.
Analysis Predicate Prompt 2. \textit{is\_endurant}: 

- The domain analyser analyses an entity, $\phi$, into an endurant as prompted by the \textit{domain analysis prompt}: 
  - \textit{is\_endurant} – $\phi$ is an endurant if $\textit{is\_endurant}(\phi)$ holds

- \textit{is\_entity} is a \textit{prerequisite prompt} for \textit{is\_endurant}.
- \textit{is\_endurant} is a method tool.
4.3.2 Perdurants

Definition 31. **Perduran**: 

- By a *perduran* we shall understand an entity 
  - for which only a fragment exists 
    if we look at or touch them 
    at any given snapshot in time. 
  - Were we to freeze time we would only see or touch 
    a fragment of the perduran.
Example 24. **Perdurants:**

**Geography Perdurants:**
- the continuous changing of the weather (meteorology);
- the erosion of coastlines;
- the rising of some land area and the “sinking” of other land area;
- volcanic eruptions;
- earthquakes;
- et cetera.

**Railway System Perdurants:**
- the ride of a train from one railway station to another; and
- the stop of a train at a railway station from some arrival time to some departure time.
Analysis Predicate Prompt 3. \textit{is\_perdurant}:

- The domain analyser analyses an entity $e$ into perdurants as prompted by the \textit{domain analysis prompt}:
  - \textit{is\_perdurant} – $e$ is a perdurant if $\text{is\_perdurant}(e)$ holds.

- \textit{is\_entity} is a prerequisite prompt for \textit{is\_perdurant}.

- \textit{is\_perdurant} is a method tool.
• We repeat method principle 3 on Slide 66:

**Method Principle 4. ** *Separation of Endurants and Perdurants:*

• First domain analyse & describe endurants;
• then domain analyse & describe perdurants
4.4 Solids and Fluids

• For pragmatic reasons we distinguish between
  – solids and
  – fluids.

Method Principle 5. Abstraction, I:

• The principle of abstraction is now brought into “full play”:
  – In analysing & describing entities the domain analyser cum
describer
  – is “free” to not consider all facets of entities,
  – that is, to abstract.
4.4.1 Solids

Definition 32. Solid Endurant::

• By a solid endurant we shall understand an endurant
• which is
  – separate,
  – individual or
  – distinct in form or concept,
• or, rephrasing:
  – a body
  – or magnitude
of three-dimensions,
having length, breadth and thickness [51, Vol. II, pg. 2046] ■
Analysis Predicate Prompt 4. \textit{is\_solid}:

- The domain analyser analyses endurants, \( e \), into solid entities as prompted by the \textit{domain analysis prompt}:
  - \textit{is\_solid} – \( e \) is solid if \( \text{is\_solid}(e) \) holds

- To simplify matters we shall allow separate elements of a solid endurant to be fluid!
- That is, a solid endurant, i.e., a part, may be conjoined with a fluid endurant, a fluid.
- \textit{is\_solid} is a method tool.
Example 25. **Artefactual Solid Endurants:**

- The individual endurants of the above example of *railway system* endurants, Example 23 on Slide 68, were all solid.

- Here are examples of solid endurants of *pipeline systems.*
  - A pipeline and
  - its individual units:

  * wells,                      * pumps,                      * regulator, and
  * valves,                    * joins,
4.4.2 **Fluids**

**Definition 33.** *Fluid Endurant:*

- By a *fluid endurant* we shall understand an endurant which is
  - prolonged, without interruption,
    in an unbroken series or pattern;

or, rephrasing:
- a substance (liquid, gas or plasma)
  having the property of flowing,
  consisting of particles
  that move among themselves [51, Vol. I, pg. 774]
Analysis Predicate Prompt 5. *is_fluid*:

- The domain analyser analyses endurants $e$ into fluid entities as prompted by the *domain analysis prompt*:
  - $is\_fluid - e$ is fluid if $is\_fluid(e)$ holds

- $is\_fluid$ is a method tool.
• Fluids are otherwise
  – liquid, or
  – gaseous, or
  – plasmatic, or
  – granular\textsuperscript{4}, or
  – plant products\textsuperscript{5},
  – et cetera.

\textsuperscript{4}This is a purely pragmatic decision.
"Of course" sand, gravel, soil, etc., are not fluids,
but for our modelling purposes it is convenient
to "compartmentalise" them as fluids!
\textsuperscript{5}i.e., chopped sugar cane, threshed, or otherwise. See footnote 4.
Example 26. **Fluids:**

- Specific examples of fluids are:
  - water, oil, gas, compressed air, etc.
- A container, which we consider a solid endurant,
  - may be *conjoined* with another, a fluid,
  - like a gas pipeline unit may “contain” gas
4.5 Parts and Living Species

• We analyse endurants into either of two kinds:
  – parts and
  – living species.

• The distinction between parts and living species
  is motivated in Kai Sørlander’s Philosophy [61, 62, 63, 64, 65].
4.5.1 Parts

Definition 34. \textit{Parts:}

- By a \textit{part} we shall understand
  - a solid endurant existing in time and
  - subject to laws of physics,
  - including the \textit{causality principle} and
  - \textit{gravitational pull}\textsuperscript{6}

\footnote{This characterisation is the result of our study of relations between philosophy and computing science, notably influenced by Kai Sørlander’s Philosphy [61, 62, 63, 64, 65]}
Analysis Predicate Prompt 6. \textit{is\_part}:

- The domain analyser analyses “things” (e) into part.
- The method can thus be said to provide the \textit{domain analysis prompt}:
  - \textit{is\_part} – where \textit{is\_part}(e) holds if e is a part

\textit{is\_part} is a method tool.
• *Parts are*
  – either *natural* parts, or are
  – *artefactual* parts, i.e. man-made.
• Natural and man-made parts are either
  – *atomic* or
  – *compound*. 
4.5.1.1 Atomic Parts

• The term ‘atomic’ is, perhaps, misleading.
  – It is not used in order to refer to nuclear physics.
  – It is, however, chosen in relation to the notion of atomism:
    * a doctrine that the physical or physical and mental universe
    * is composed of simple indivisible minute particles [Merriam Webster].

Definition 35. Atomic Part, II:

• By an atomic part we shall understand a part
  – which the domain analyser considers to be indivisible
  – in the sense of not meaningfully,
  – for the purposes of the domain under consideration,
  – that is, to not meaningfully consist of sub-parts.
Example 27. **Atomic Parts:**

- We refer to Example ?? on Slide ??: pipeline systems. The
  - wells,
  - pumps,
  - valves,
  - pipes,
  - forks,
  - joins and
  - sinks

  can be considered atomic
Analysis Predicate Prompt 7. *is_atomic*:

- The domain analyser analyses “things” (e) into atomic part.
- The method can thus be said to provide
  
  the *domain analysis prompt*:

  - *is_atomic* – where *is_atomic(e)* holds if e is an atomic part.

*is_atomic* is a method tool.
4.5.1.2 Compound Parts, II

• We, pragmatically, distinguish between
  – Cartesian-product-, and
  – set-oriented parts.

• That is, if Cartesian-product-oriented, to consist of two or more distinctly sort-named endurants (solids or fluids),

• or, if set-oriented, to consist of an indefinite number of zero, one or more identically sort-named parts.
Definition 36. *Compound Part:*

- *Compound part* s are those which are
  - either Cartesian-product-
  - or are set-

- oriented parts ■
Analysis Predicate Prompt 8. *is\_compound*:

- The domain analyser analyses “things” \((e)\) into compound part.
- The method can thus be said to provide the domain analysis prompt:
  
  – *is\_compound* – where *is\_compound*\((e)\) holds if \(e\) is a compound part

*is\_compound* is a method tool.
4.5.1.2.1 Cartesian Parts

Definition 37. *Cartesian Part:*

- *Cartesian parts* are those (compound parts)
  - which consists of an “indefinite number”
  - of two or more parts
  - of distinctly named sorts
Example 28. Cartesian Automobiles:

- We refer to Example 23 on Slide 69, the transport system sub-example.
- We there viewed (hubs, links and) automobiles as atomic parts.
- From another point of view we shall here understand automobiles as Cartesian parts:
  - the engine train,
  - the chassis,
  - the car body,
  - four doors (left front, left rear, right front, right rear), and
  - the wheels.
- These may again be considered Cartesian parts.
Analysis Predicate Prompt 9. *is_Cartesian*:

- The domain analyser analyses “things” (e) into Cartesian part.
- The method can thus be said to provide the *domain analysis prompt*:
  - *is_Cartesian* – where *is_Cartesian(e)* holds if e is a Cartesian part.

is_Cartesian is a method tool.
4.5.1.2.2 Calculating Cartesian Part Sorts

• The above analysis amounts to the analyser
  – first “applying” the domain analysis prompt
  – is_compound(e) to a solid endurant, e,
  – where we now assume that the obtained truth value is true.
  – Let us assume that endurants e:E consist of sub-endurants of sorts
    \{E_1,E_2,\ldots,E_m\}.
  – Since we cannot automatically guarantee that our domain descriptions secure that
  – E and each E_i (1\leq i\leq m)
  – denotes disjoint sets of entities we must prove so!
On Determination Functions:

• Determination functions
  – apply to compound parts
  – and yield their sub-parts and the sorts of these.

• *That is,*
  – *we observe the domain*
  – *and our observation results*
  – *in a focus on a subset of that domain*
  – *and sort information about that subset.*
An RSL Extension:

• The `determine` functions below are expressed as follows:
  
  `value determine_····(e) as (parts,sorts)`

  – where we focus here on the sorts clause.
  – Typically that clause is of the form \( \eta A, \eta B,..., \eta C \).
  – That is, a “pattern” of sort names: A,B,...,C.

• These sort names are provided by the domain analyser cum describer.

• They are chosen as “full names”, or as mnemonics, to capture an essence of the (to be) described sort.

• Repeated invocations, by the domain analyser cum describer, of these (... , sorts) analysis functions normally lead to new sort names distinct from previously chosen such names.
4.5.1.2.2.1 Cartesian Part Determination

Observer Function Prompt 1. \textit{determine Cartesian parts:}

- The domain analyser analyses a part into a Cartesian part.
- The method provides the \textit{domain observer prompt}:
  - \textit{determine Cartesian parts} — it directs the domain analyser to determine the definite number of values and corresponding distinct sorts of the part.

\text{value}

\text{determine Cartesian parts: } E \rightarrow (E_1 \times E_2 \times \ldots \times E_n) \times (\eta E_1 \times \eta E_2 \times \ldots \times \eta E_n)^8
\text{determine Cartesian parts}(e) \text{ as } ((e_1, \ldots, e_n), (\eta E_1, \ldots, \eta E_n))

where by \( E, E_i \) we mean endurants, i.e., part values, and by \( \eta E_i \) we mean the names of the corresponding types.

\textit{determine Cartesian parts} is a method tool.

\textsuperscript{7} \eta A, \eta B, \ldots, \eta C are \textbf{names} of types. \eta \theta \textbf{ is the type of all type names}!
\textsuperscript{6} The ordering, \(((e_1, \ldots, e_n), (\eta E_1, \ldots, \eta E_n))\), is pairwise arbitrary.
On Calculate Prompts:

- Calculation prompts
  - apply to compound parts: Cartesians and sets,
  - and yield an RSL-Text description.
Domain Description Prompt 2. *calc_Cartesian_parts:*

- *If is_Cartesian(e) holds, then the analyser “applies” the domain description prompt*
  
  – *calc_Cartesian_parts(e)*

resulting in the analyser writing down

the endurant sorts and endurant sort observers
domain description text
according to the following schema:
**calc_Cartesian_parts** *descriptor*

let \((\_^9,(\eta E_1,\ldots,\eta E_m))) = \text{determine}\_\text{Cartesian}\_\text{parts}\_\text{sorts}\_\text{sorts}(e)^{10}\text{ in}

**Narration:**

[s] ... narrative text on sorts ...
[o] ... narrative text on sort observers ...
[p] ... narrative text on proof obligations ...

**Formalisation:**

type
[s] \(E_1, \ldots, E_m\)

value
[o] \(\text{obs}_{E_1} : E \to E_1, \ldots, \text{obs}_{E_m} : E \to E_m\)

proof obligation
[p] [Disjointness of endurant sorts]
calc_Cartesian_parts is a method tool.

*The use of the underscore, _, shall inform the reader that there is no need, here, for naming a value.

*For determine_composite_parts see Sect. 4.5.1.2.2.1 on Slide 99
Elaboration 1  *Type, Values and Type Names:*

- *Note the use of quotes above.*
- *Please observe that when we write obs_E then obs_E is the name of a function.*
- *The E, when juxtaposed to obs_ is now a name*
Observer Function Prompt 2. `type_name, type_of`:
The definition of `type_name, type_of` implies the informal definition of

\[
\text{obs}_{E_i}(e) = e_i \equiv \text{type}_\text{name}(e_i) = "E_i" \land \\
\text{type}_\text{of}(e_i) = E_i \land \\
\text{is}_{E_i}(e_i)
\]
Example 29. A Road Transport System Domain: Cartesians:

14. There is the *universe of discourse*, RTS.
   
   It is composed from

15. a *road net*, RN, and

16. an *aggregate of automobiles*, AA.

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>obs_RN: RTS → RN</td>
</tr>
<tr>
<td>RN</td>
<td></td>
</tr>
<tr>
<td>AA</td>
<td>obs_AA: RTS → AA</td>
</tr>
</tbody>
</table>
• We continue the analysis & description of “our” road transport system:

17. The road net consists of
   (a) an aggregate, AH, of hubs and
   (b) an aggregate, AL, of links.

   **type**
   17a  AH
   17b  AL

   **value**
   17a  obs_AH: RN → AH
   17b  obs_AL: RN → AL
4.5.1.2.3 Part Sets

Definition 38. Part Sets:

- Part sets are those which,
  - in a given context,
  - are deemed to meaningfully consist of separately observable
    * a [“root”] part and
    * an indefinite number of proper [“sibling”] sub-parts

- For pragmatic reasons we distinguish between parts sets all of whose parts are
  - of the same, single, further un-analysed sort, and
  - of two or more distinct atomic sorts.
Definition 39. *Single Sort Part Sets:*

- *Single sort part sets* are those which,
  - in a given context,
  - are deemed to *meaningfully* consist of separately observable
    - a ["root"] part and
    - an indefinite number of proper ["sibling"] 'sub-parts of the same, i.e., single sort.
Analysis Predicate Prompt 10. \textit{is\_single\_sort\_set}:

- The domain analyser analyses a solid endurant, i.e., a part $p$ into a set endurant:
  
  - $\textit{is\_single\_sort\_set}$: $p$ is a composite endurant
    
    if $\textit{is\_single\_sort\_set}(p)$ holds

$\textit{is\_single\_sort\_set}$ is a method tool.
• The `is_single_sort_set` predicate is informal.
• So are all the domain analysis predicates (and functions).
• That is,
  – Their values are “calculated” by a human, the domain analyser.
  – That person observes parts in the “real world”.
  – The determination of the predicate values, hence, are subjective.
Definition 40. *Alternative Atomic Part Sets:*

- *Alternative sorts part set* s are those which,
  - in a given context,
  - are deemed to *meaningfully* consist of separately observable
    * a [“root”] part and
    * an indefinite number of proper [“sibling”] `sub-parts of two or more atomic parts of distinct sorts`
Analysis Predicate Prompt 11. \textit{is\textunderscore alternative\textunderscore sorts\textunderscore set}:

- The domain analyser analyses a solid endurant, i.e., a part \( p \) into a set endurant:
  
  - \textit{is\textunderscore alternative\textunderscore sorts\textunderscore set}: \( p \) is a composite endurant if \( \text{is\textunderscore alternative\textunderscore sorts\textunderscore set}(p) \) holds.

\textit{is\textunderscore alternative\textunderscore sorts\textunderscore set} is a method tool.
4.5.1.2.3.1 Determine Same Sort Part Sets

Observer Function Prompt 3. *determine_same_sort_parts_set:*

- The domain analyser observes parts into same sorts part sets.
- The method provides the domain observer prompt:
  - *determine_alternative_sorts_part_set* directs the domain analyser to determine the values and corresponding sorts of the part.

\[
\text{value } \quad \text{determine_same_sort_part_set: } E \rightarrow (P\text{-set} \times \theta P) \\
\text{determine_same_sort_part_set(e) as } (ps, \eta Pn)
\]

*determine_same_sort_part_set* is a method tool.
4.5.1.2.3.2 Determine Alternative Sorts Part Sets

Observer Function Prompt 4.

determine_alternative_sorts_part_set:

- The domain analyser observes parts into alternative sorts part sets.
- The method provides the domain observer prompt:
  - determine_alternative_sorts_part_set directs the domain analyser to determine the values and corresponding sorts of the part.

  value
determine_alternative_sorts_part_set: E → ((P1×θP1)×...×(Pn,θPn))
determine_alternative_sorts_part_set(e) as ((p1,ηp1),..., (pn,ηPn))

- The set of parts, of different sorts, may have more than one element, p, p′,..., p” being of the same sort Ei.

determine_alternative_sorts_part_set is a method tool.
4.5.1.2.3.3 Calculating Single Sort Part Sets

Domain Description Prompt 3. \texttt{calc\_single\_sort\_parts\_sort}:

• If \texttt{is\_single\_set\_sort\_parts}(e) holds, then the analyser “applies” the domain description prompt
  \newline
  – \texttt{calc\_single\_sort\_parts}(e)

resulting in the analyser writing down the single set sort and sort observers domain description text according to the following schema:
**calculate_single_sort_parts_sort(e)** Describer

```
let (__, η | P) = determine_single_sort_part(e)\textsuperscript{11} in 
```

“Narration:
[s] ... narrative text on sort ...
[o] ... narrative text on sort observer ...
[p] ... narrative text on proof obligation ...

**Formalisation:**

```
type 
[s] P
[s] Ps = P-set
value 
[o] obs_Ps: E → Ps 
end
```

calculate_single_sort_parts_sort is a method tool.

\textsuperscript{11}For determine_single_sort_part see Defn. 39 on Slide 109.
Elaboration 2 *Type, Values and Type Names:*

- *Note the use of quotes above.*
- *Please observe that when we write obs_Ps then obs_Ps is the name of a function.*
- *The Ps, when juxtaposed to obs_is now a name*
Example 30. **Road Transport System: Sets of Hubs, Links and Automobiles:** We refer to Example 29 on Slide 106.

18. The road net aggregate of road net hubs consists of a set of [atomic] hubs,

19. The road net aggregate of road net links consists of a set of [atomic] links,

20. The road net aggregate of automobiles consists of a set of [atomic] automobiles.

**type**

18. $Hs = H\text{-set}, H$
18. $Ls = L\text{-set}, L$
18. $As = A\text{-set}, A$

**value**

18. $\text{obs}_Hs: AH \rightarrow Hs$
18. $\text{obs}_Ls: AL \rightarrow Ls$
18. $\text{obs}_As: AA \rightarrow As$
4.5.1.2.3.4 Calculating Alternative Sort Part Sets

- We leave it to the reader to decipher the calculate_alternative_sort_part_sorts prompt.

Domain Description Prompt 4.
calculate_alternative_sort_part_sorts:

- If is_alternative_sort_parts_sorts(e) holds, then the analyser “applies” the domain description prompt
  resulting in the analyser writing down the alternative sort and sort observers domain description text according to the following schema:
calculate_alternative_sort_part_sorts(e) Describer

let ((p1,ηE_1),...,(pn,ηE_n)) = determine_alternative_sorts_part_set_sorts(e)_{12} in

“Narration:
[s] ... narrative text on alternative sorts ... 
[o] ... narrative text on sort observers ... 
[p] ... narrative text on proof obligations ... 

Formalisation:

type
[s] Ea = E_1 | ... | E_n 
[s] E_1 :: End_1, ..., E_n :: End_n 

value
[o] obs_Ea: E → Ea 

axiom
[p] [ disjointness of alternative sorts ] E_1, ..., E_n ”

end
• The set of parts, of different sorts, may have more than one element, say \( p, p', ..., p'' \) being of the same sort \( E_i \).
  – Since parts are not mentioned in the sort description above, cf.,
  – only the distinct alternative sort observers appear in that description.

\texttt{calculate\_alternative\_sort\_part\_sorts} is a method tool.

\footnote{For \texttt{determine\_alternative\_sort\_part\_sorts} see Defn. 40 on Slide 112.}
Example 31. Alternative Rail Units:

21. The example is that of a railway system.

22. We focus on railway nets. They can be observed from the railway system.

23. The railway net embodies a set of [railway] net units.

24. A net unit is either a

   • straight or curved linear unit, or a
   • simple switch, i.e., a turnout, unit\textsuperscript{13} or
   • a simple cross-over, i.e., a rigid crossing unit, or a
   • single switched cross-over, i.e., a single slip unit, or a
   • double switched cross-over, i.e., a double slip unit, or a
   • terminal unit.

25. As a formal specification language technicality disjointness of the respective rail unit types is afforded by RSL’s :: type definition construct.
• We refer to Figure 4.2 on the next slide.

**type**

21. RS
22. RN

**value**

22. obs\_RN: RS → RN

**type**

23. NUs = NU-set
24. NU = LU|PU|RU|SU|DU|TU

25. LU :: LinU
25. PU :: PntU
25. SU :: SwiU
25. DU :: DblU
25. TU :: TerU

23. obs\_NUs: RN → NUs

---

Figure 4.2: Left: Four net units (LU, PU, SU, DU); Right: A railway net

- This section, i.e., Sect. 4.5.1, has illustrated a principle of “small, pedantic” analysis & description steps.
  - You could also call it a principle of separation of concerns
4.5.1.3 Ontology and Taxonomy

• We can speak of two kinds of ontologies:
  – the general ontologies of domain analysis & description, cf. Fig. 4.1 on Slide 63, and
  – a specific domain’s possible endurant ontologies.
– We shall here focus on a [“restricted”] concept of taxonomies\(^{14}\)

\(^{14}\)By taxonomy (or taxonomical classification) we shall here understand a scheme of classification, especially a hierarchical classification, in which things are organized into groups [Wikipedia].
**Definition 41.** *Domain Taxonomy:* By a domain taxonomy we shall understand

- a hierarchical structure, usually depicted as a(n “upside-down”) tree,
- whose “root” designates a compound part
- and whose “siblings” (proper sub-trees) designate parts or fluids

- The ‘restriction’ amounts to considering only endurants.
- That is, not considering perdurants.
- *Taxonomy* is a method technique.
Example 32. The Road Transport System Taxonomy:

- Figure 4.3 shows a schematised, i.e., the ..., taxonomy for the Road Transport System domain of Example 4.1 on Slide 63.

Figure 4.3: A Road Transport System Ontology
4.5.1.4 “Root” and “Sibling” Parts

- For compound parts, cf. Definition 36 on Slide 91,
  - we introduce the specific domain taxonomy concepts of “root” and “sibling” parts.
  - (We also refer to Fig. 4.3 on the preceding slide.)

- When observing, as a human, a compound part one may ask the question
  - “a tree consisting of a specific domain taxonomy node labelled, e.g., $X$
  - and the sub-trees labelled, e.g., $Y_1, Y_2, \ldots, Y_n$
  - does that tree designate one “indivisible” part
  - or does it designate $n + 1$ parts?”
  - We shall, in general, consider the answer to be the latter: $n + 1$!
• We shall, in general, consider compound parts to consist of
  – a “root” parts
  – and $n$ “sibling parts and fluids”.

• What the analyser cum describer observes
  – appears as one part, “the whole”,
  – with $n$ “embedded” sub-parts.

• What the analyser cum describer is asked to model is
  – 1, the root part, and
  – $n$, the sibling, parts and fluids.
• The fact that the root part is separately modelled from the sibling parts,
• may seem to disappear in this separate modelling —
• but, as You shall see, in the next chapter,
  – their relation: the siblings to “the whole”, i.e., the root,
  – will be modelled, specifically through their mereologies,
  – as will be covered in Sect. 5.3,
  – but also through their respective attributes, Sect. 5.4.
• We shall see this non-embbededness of root and sibling parts
  – further accentuated in the modelling of their transcendentally deduced
  – respective (perdurant) behaviours as distinct concurrent behaviours
  – in Chapter 6.
4.5.2 Living Species

• *Living Species* are
  – either *plants*
  – or *animals*.

• Among animals we have the *humans*.
Definition 42. *Living Species:*

- By a *living species* we shall understand
  - a solid endurant,
  - subject to laws of physics, and
  - additionally subject to *causality of purpose*.

- *Living species*
  - must have some *form they can be developed to reach*;
  - a form they must be *causally determined to maintain*.
  - This *development and maintenance* must further
    engage in *exchanges of matter with an environment*.

- It must be possible that living species occur in two forms:
  - *plants*, respectively *animals*,
  - forms which are characterised by
    *development, form and exchange*,
  - which, additionally, can be characterised by
    the *ability of purposeful movement*.
Analysis Predicate Prompt 12. \textit{is\_living\_species}: 

- The domain analyser analyses “things” \((e)\) into living species.
- The method can thus be said to provide the \textit{domain analysis prompt}:
  - \textit{is\_living\_species} – where \textit{is\_living\_species}(\(e\)) holds if \(e\) is a living species

\textit{is\_living\_species} is a method tool.
• It is appropriate here to mention Carl Linnaeus (1707–1778).
  – He was a Swedish botanist, zoologist, and physician
  – who formalised, in the form of a binomial nomenclature,
  – the modern system of naming organisms.
  – He is known as the “father of modern taxonomy”.
  – We refer to his ‘Species Plantarum’
    gutenberg.org/files/20771/20771-h/20771-h.htm.
4.5.2.1 Plants

Example 33. Plants:

• Although we have not yet come across domains for which the need to model the living species of plants were needed, we give some examples anyway:
  – grass,
  – tulip,
  – rhododendron,
  – oak tree.
Analysis Predicate Prompt 13. *is_plant*:

- The domain analyser analyses “things” (ℓ) into a plant.
- The method can thus be said to provide the domain analysis prompt:
  - *is_plant* – where *is_plant*(ℓ) holds if ℓ is a plant

*is_plant* is a method tool.

- The predicate *is_living_species*(ℓ) is a prerequisite for *is_plant*(ℓ).
4.5.2.2 Animals

Definition 43. *Animal:* We refer to the initial definition of *living species* above – while emphasizing the following traits:

- (i) *a form that animals can be developed to reach* and
- (ii) *causally determined to maintain* through
- (iii) *development and maintenance* in an *exchange of matter with an environment*, and
- (iv) *ability to purposeful movement*
Analysis Predicate Prompt 14. \textit{is\_animal}:

- The domain analyser analyses “things” (ℓ) into an animal.
- The method can thus be said to provide the \textit{domain analysis prompt}:
  - \textit{is\_animal} – where \textit{is\_animal}(ℓ) holds
    - if ℓ is an animal
- \textit{is\_animal} is a method tool.
- The predicate \textit{is\_living\_species}(ℓ) is a prerequisite for \textit{is\_animal}(ℓ).
- We distinguish, motivated by [64], between
  - humans and
  - other.
4.5.2.2.1 Humans

Definition 44. **Human:**

- A *human* (a *person*) is an *animal*, cf. Definition 43 on Slide 139, with the additional properties of having
  - *language*,
  - being *conscious* of having *knowledge* (of its own situation), and
  - *responsibility*
Analysis Predicate Prompt 15. \textit{is\_human}:

- The domain analyser analyses “things” (\(\ell\)) into a human.
- The method can thus be said to provide the domain analysis prompt:
  - \textit{is\_human} – where \textit{is\_human}(\(\ell\)) holds if \(\ell\) is a human
- \textit{is\_human} is a method tool.
- The predicate \textit{is\_animal}(\(\ell\)) is a prerequisite for \textit{is\_human}(\(\ell\)).
• We have not, in our many experimental domain modelling efforts
  – had occasion to model humans;
  – or rather:
    * we have modelled, for example, automobiles
      · as possessing human qualities,
      · i.e., “subsuming humans”.

© Dines Bjørner, September 8, 2022
The TU Wien Lectures, Fall 2022
• We have found, in these experimental domain modelling efforts
  – that we often confer anthropomorphic qualities on artefacts,
  – that is, that these artefacts have human characteristics.
• You, the listeners, are reminded
  – that when some programmers try to explain their programs
  – they do so using such phrases as
  – *and here the program does ... so-and-so!*
4.5.2.2.2 Other

- We shall skip any treatment of other than human animals!
4.6 Some Observations

- Two observations must be made.
  - (i) The domain analyser cum describer procedures
    - illustrated by the analysis functions
    - `determine_Cartesian_parts`,
    - `determine_same_sort_part_set` and
    - `determine_alternative_sorts_part_set`
    - yield names of endurant sorts.
    - Some of these names may have already been encountered, i.e., discovered.
    - That is, the domain analyser cum describer must carefully consider such possibilities.
– (ii) Endurants are not recursively definable!
  * This appears to come as a surprise to many computer scientists.
  * Immediately many suggest that “tree-like” endurants like a river,
  * or, indeed, a tree,
  * should be defined recursively.
  * But we posit that that is not the case.
  * A river, for example, has a delta, its “root” so-to-speak,
  * but the sub-trees of a recursively defined river endurant has no such “deltas”!
  * Instead we define such “tree-like” endurants as graphs with appropriate mereologies.
4.7 States

• In our continued modelling
  – we shall make good use of a concept of states.

Definition 45. **State:** By a state we shall understand
• any collection of one or more parts
• In Chapter 5 Sect. 5.4 we introduce the notion of attributes.
  – Among attributes there are the *dynamic attributes*.
  – They model that internal part quality values may change dynamically.
  – So we may wish, on occasion, to ‘refine’ our notion of state to be just those parts which have dynamic attributes.
4.7.1 State Calculation

- Given any universe of discourse, \( uod:UoD \), we can recursively calculate its “full” state, \( calc\_parts(\{uod\}) \).

26. Let \( e \) be any endurant. Let \( arg\_parts \) be the parts to be calculated. Let \( res\_parts \) be the parts calculated. Initialise the calculator with \( arg\_parts=\{e\} \) and \( res\_parts=\{} \). Calculation stops with \( arg\_parts \) empty and \( res\_parts \) the result.

27. If \( is\_Cartesian(e) \)
28. then we obtain its immediate parts, \( determine\_composite\_part(e) \)
29. add them, as a set, to \( arg\_parts \), \( e \) removed from \( arg\_parts \) and added to \( res\_parts \) calculating the parts from that.

30. If \( is\_single\_sort\_part\_set(e) \)
31. then the parts, \( ps \), of the single sort set are determined,
32. added to \( arg\_parts \) and \( e \) removed from \( arg\_parts \) and added to \( res\_parts \) calculating the parts from that.

33. If \( is\_alternative\_sorts\_part\_set(e) \) then the parts, \( ((p_1), (p_2), ..., (p_n)) \), of the alternative sorts set are determined, added to \( arg\_parts \) and \( e \) removed from \( arg\_parts \) and added to \( res\_parts \) calculating the parts from that.
value
26. calc_parts: E-set → E-set → E-set
26. calc_parts(arg_parts)(res_parts) ≡
26. if arg_parts = {} then res_parts else
26. let e \in arg_parts in
27. is_Cartesian(e) →
28. let ((e1,e2,...,en),__) = observe_Cartesian_part(e) in
29. calc_parts(arg_parts\{e\} \cup \{e1,e2,...,en\})(res_parts \cup \{e\}) end
30. is_single_sort_part_set(e) →
31. let ps = observe_single_sort_part_set(e) in
32. calc_parts(arg_parts\{e\} \cup ps)(res_parts \cup \{e\}) end
33. is_alternative_sort_part_set(e) →
33. let ((p1,__),(p2,__),...,__(pn,__)) = observe_alternative_sorts_part_set(e) in
33. calc_parts(arg_parts\{e\} \cup \{p1,p2,...,pn\})(res_parts \cup \{e\}) end
26. end end

calc_parts is a method tool.
Method Principle 7. **Domain State:**

- *We have found, once all the state components, i.e., the endurant parts, have had their external qualities analysed, that*
- *it is then expedient to define the domain state.*
- *It can then be the basis for several concepts*
- *of internal qualities.*
Example 34. **Constants and States:**

34. Let there be given a universe of discourse, $rts$. The set \{rts\} is an example of a state.

From that state we can calculate other states.

35. The set of all hubs, $hs$.
36. The set of all links, $ls$.
37. The set of all hubs and links, $hls$.
38. The set of all automobiles, $as$.
39. The set of all parts, $ps$. 
value

34  \( rts:UoD \quad [34] \)
35  \( hs:H-set \equiv \text{obs}_{SH}(\text{obs}_{SH}(\text{obs}_{RN}(rts))) \)
36  \( ls:L-set \equiv \text{obs}_{SL}(\text{obs}_{SL}(\text{obs}_{RN}(rts))) \)
37  \( hls:(H|L)-set \equiv hs \cup ls \)
38  \( as:A-set \equiv \text{obs}_{As}(\text{obs}_{AA}(\text{obs}_{RN}(rts))) \)
39  \( ps:(UoB|H|L|A)-set \equiv rts \cup hls \cup as \)
4.7.2 Update-able States

• We shall, in Sect. 5.4, introduce the notion of parts,
  – having dynamic attributes,
  – that is, having internal qualities that may change.
• To cope with the modelling,
• in particular of so-called *monitor-able* attributes,
• we present the state as a global variable:

```
variable \( \sigma := \text{calc\_parts}\{\text{uod}\} \)
```
4.8 An External Analysis and Description Procedure

• We have covered
  – the individual analysis and description steps
  – of our approach to the external qualities modelling
  – of domain endurants.

• We now suggest
  – a ‘formal’ description of the process
  – of linking all these analysis and description steps.
4.8.1 An Analysis & Description State

• Common to all the discovery processes is an idea of a notice board.

• A notice board, at any time in the development of a domain description, is a repository of the analysis and description process.

• We suggest to model the notice board in terms of three global variables.
  – The new variable holds the parts yet to be described,
  – The ans variable holds the sort name of parts that have so far been described,
  – the gen variable holds the parts that have so far been described, and
  – the txt variable holds the RSL-Text so far generated.

  ∗ We model the txt variable as a map
  ∗ from endurant identifier names to RSL-Text.
A Domain Discovery Notice Board

variable
new := \{uod\} ,
asn := \{ “UoD” \}
gen := \{ \},
txt:RSL-Text := [ uid_UoD(uod) \mapsto \langle “type UoD” \rangle ]
4.8.2  A Domain Discovery Procedure, I

• The discover_sorts pseudo program
  – suggests a systematic way of proceeding
  – through analysis, manifested by the is_... predicates,
  – to (→) description.

• Some comments are in order.
  – The e-set_a∪e-set_b expression
  – yields a set of endurants that are either in e-set_a, or in e-set_b, or in both,
  – but such that two endurants, e_x and e_y
  – which are of the same endurants type, say E,
  – and are in respective sets is only represented once in the result;
  – that is, if they are type-wise the same, but value-wise different
  – they will only be included once in the result.

• As this is the first time RSL-Text is put on the notice board we express this as:
  – txt := txt ∪ [type_name(v) ↦ ⟨RSL-Text⟩]

• Subsequent insertion of RSL-Text
  for internal quality descriptions and perdurants
  is then concatenated to the end of previously uploaded RSL-Text.
An External Qualities Domain Analysis and Description Process

value
discover-sorts: $\text{Unit} \rightarrow \text{Unit}$
discover-sorts() ≡ while new ≠ {} do
  let $v \cdot v \in \text{new}$ in (new := new \ {v} || gen := gen \cup \{v\} || ans := ans \ \{\text{type_of}(v)\}) ;
  is_atomic(v) → skip ,
  is_compound(v) → is_Cartesian(v) →
  let ((e1, ..., en), (ηE1, ..., ηEn)) = analyse_composite_parts(v) in
  (ans := ans \cup \{ηE1, ..., ηEn\} || new := new \{e1, ..., en\}
    || txt := txt \cup [\text{type_name}(v) \mapsto \langle\text{calculate_composite_part_sorts}(v)\rangle]) end,
  is_part_set(v) →
  (is_single_sort_set(v) →
   let ([p1, ..., pn], ηP) = analyse_single_sort_parts_set(v) in
   (ans := ans \cup [ηP] || new := new \{p1, ..., pn\} ||
     txt := txt \cup [\text{type_name}(v) \mapsto \text{calculate_single_sort_part_sort}(v)]) end,
  is_alternative_sorts_set(v) →
   let ((p1, ηE1), ..., (pn, ηEn)) = observe_alternative_sorts_part_set(v) in
   (ans := ans \cup [ηE1, ..., ηEn] || new := new \{p1, ..., pn\} ||
     txt := txt \cup [\text{type_name}(v) \mapsto \text{calculate_alternative_sorts_part_sort}(v)]) end)
end end
4.9 **Summary**

- We briefly summarise the main findings of this chapter.
- These are the main
  - analysis predicates and functions and
  - the main description functions.
- These, to remind the student, are
  - the *analysis*, the *is_*..., *predicates*,
  - the *analysis*, the *determine_*..., *functions*,
  - the *state calculation* function,
  - the *description* functions, and
  - the *domain discovery* procedure.
• They are summarised in this table:

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Introduced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis Predicates</td>
<td>1 is_entity</td>
<td>page 60</td>
</tr>
<tr>
<td></td>
<td>2 is_endurant</td>
<td>page 70</td>
</tr>
<tr>
<td></td>
<td>3 is_perdurant</td>
<td>page 73</td>
</tr>
<tr>
<td></td>
<td>4 is_solid</td>
<td>page 77</td>
</tr>
<tr>
<td></td>
<td>5 is_fluid</td>
<td>page 80</td>
</tr>
<tr>
<td></td>
<td>6 is_part</td>
<td>page 85</td>
</tr>
<tr>
<td></td>
<td>7 is_atomic</td>
<td>page 89</td>
</tr>
<tr>
<td></td>
<td>8 is_compound</td>
<td>page 92</td>
</tr>
<tr>
<td></td>
<td>9 is_Cartesian</td>
<td>page 95</td>
</tr>
<tr>
<td></td>
<td>10 is_single_sort_set</td>
<td>page 110</td>
</tr>
<tr>
<td></td>
<td>11 is_alternative_sorts_set</td>
<td>page 113</td>
</tr>
<tr>
<td></td>
<td>12 is_living_species</td>
<td>page 135</td>
</tr>
<tr>
<td></td>
<td>13 is_plant</td>
<td>page 138</td>
</tr>
<tr>
<td></td>
<td>14 is_animal</td>
<td>page 140</td>
</tr>
<tr>
<td></td>
<td>15 is_human</td>
<td>page 142</td>
</tr>
</tbody>
</table>

| Analysis Functions | 1 determine_Cartesian_parts | page 99 |
| | 3 determine_same_sort_part_set | page 114 |
| | 4 determine_alternative_sorts_part_set | page 115 |

| State Calculation | calc_parts | page 150 |

| Description Functions | 1 calc_Universe_of_Discourse | page 55 |
| | 2 calc_Cartesian_parts | page 101 |
| | 3 calc_single_sort_parts_sort | page 116 |
| | 4 calc_alternative_sort_part_sorts | page 117 |

| Domain Discovery | discover_sorts | page 160 |
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.
CHAPTER 5. Endurants: Internal Domain Qualities

• In this chapter we introduce
  – the concept of internal qualities of endurants,
  – and cover the analysis and description of
    ∗ unique identifiers,
    ∗ mereologies and
    ∗ attributes
    of endurants.
  – There is yet another, interrelating internal quality:
    ∗ intentionality, “something” that expresses
    ∗ intention, design idea, purpose of artefacts –
    ∗ well, some would say, also 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 [24] 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. \textit{is\_manifest}:

- The method provides the \textbf{domain analysis prompt}:
  - \textit{is\_manifest} – where \textit{is\_manifest}(p)
    holds if \textit{p} is to be considered manifest.

Analysis Predicate Prompt 17. \textit{is\_structure}:

- The method provides the \textbf{domain analysis prompt}:
  - \textit{is\_structure} – where \textit{is\_structure}(p)
    holds if \textit{p} is to be considered a structure.

- The obvious holds: \textit{is\_manifest}(p) \equiv \neg \textit{is\_structure}(p).
5.1.2.3 Examples

Example 35. Manifest Parts and Structures:
We refer to Example 29 on Slide 106: 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 identifiers,
  – (ii) that unique identifiers (of part endurants e:E) are abstract values
    (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, and
  – (iv) that all ui_i:UI_i and ui_j:UI_j are distinct.

\[\footnotesize{\text{\textsuperscript{5}}This restriction is not necessary, but, for the time, we can assume that it is.}\]
• The names of unique identifier sorts, say Ul, 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 Pl.
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 Philosophy,
  – 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 identifiability 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:Ul, of e.

Domain Description Prompt 5. \textit{describe\textunderscore unique\textunderscore identifier}(e):

• We can therefore apply the \textit{domain description prompt}:
  – \textit{describe\textunderscore unique\textunderscore 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 ...<s>\(^7\)
<s>... narrative text on unique identifier observer uid_E ... <s>
<s>... axiom on uniqueness of unique identifiers ...<s>

**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*
  
  $\forall e \in E \cdot \text{uid}_E(e) = ui \Rightarrow$
  
  $\text{type_of}\left(\text{ui}\right) = \eta_U^{\text{I}} \land \text{type_name}(\text{ui}) = \text{UI} \land \text{is}_\text{UI}(\text{ui})$

- $\eta_U^{\text{I}}$ is a variable of type $\eta^T$.

- $\eta^T$ 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 $\eta_U^{\text{I}}$. 
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  \( uid_{H}: H \rightarrow H_{UI} \)

42b  \( uid_{L}: H \rightarrow L_{UI} \)

42c  \( uid_{A}: H \rightarrow A_{UI} \)
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.

\[
\text{calculate all unique identifiers: } \text{UoD} \rightarrow \text{UI-set}
\]

\[
\text{calculate all unique identifiers(uod) } \equiv
\]

\[
\text{let parts = calc_parts({uod})({}) in}
\]

\[
\{ \text{uid}_E(e) \mid e:E \cdot e \in \text{parts} \} \text{ end}
\]
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 set, $h_{u,i}$, of unique hub identifiers;
44. the set, $l_{u,i}$, of unique link identifiers;
45. the set, $r_{u,i}$, of all unique hub and link, i.e., road identifiers;
46. the map, $hl_{u,i}$, from unique hub identifiers to the set of unique link identifiers of the links connected to the zero, one or more identified hubs,
47. the map, $lh_{u,i}$, from unique link identifiers to the set of unique hub identifiers of the two hubs connected to the identified link;
48. the set, $a_{u,i}$, of unique automobile identifiers;
value
43 \( h_{ui}:\text{H\_Ul-set} \equiv \{\text{uid}_H(h)|h:H; h \in hs\} \)
44 \( l_{ui}:\text{L\_Ul-set} \equiv \{\text{uid}_L(l)|l:L; l \in ls\} \)
45 \( r_{ui}:\text{R\_Ul-set} \equiv h_{ui} \cup l_{ui} \)
46 \( hl_{ui}: (\text{H\_Ul} \rightarrow \text{L\_Ul-set}) \equiv \)
46 \[[ h_{ui} \mapsto \text{luis}|h_{ui}:\text{H\_Ul},\text{luis}:\text{L\_Ul-set}: h_{ui} \in h_{ui}s \land (\_,\text{luis},\_)=\text{mero}_H(\eta(h_{ui})) \]
47 \( lh_{ui}: (\text{L\_Ul} \rightarrow \text{H\_Ul-set}) \equiv \)
47 \[[ l_{ui} \mapsto \text{huis}|h_{ui}:\text{L\_Ul},\text{huis}:\text{H\_Ul-set} \cdot l_{ui} \in l_{ui}s \land (\_,\text{huis},\_)=\text{mero}_L(\eta(l_{ui})) \]
48 \( a_{ui}:\text{A\_Ul-set} \equiv \{\text{uid}_A(a)|a:A; a \in as\} \)
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 \( \text{card} \ \text{calc_parts}([\text{uod}]) = \text{card} \ \text{all_uniq_ids}() \)
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. \( \text{card } h_s = \text{card } h_{ui}s \)
51. \( \text{card } l_s = \text{card } l_{ui}s \)
52. \( \text{card } a_s = \text{card } a_{ui}s \)
53. \( \text{card } \{h_{ui}s \cup l_{ui}s \cup b_{ui}s \cup b_{ui}s \cup a_{ui}s\} \)
53. \( \text{card } h_{ui}s + \text{card } l_{ui}s + \text{card } b_{ui}s + \text{card } b_{ui}s + \text{card } a_{ui}s \)
• 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.

\[ \text{type} \]
54. \( UI \)

\[ \text{value} \]
54. \( \text{uid}_\text{NU}: \text{NU} \rightarrow UI \)

\[ \text{axiom} \]
55. \( \forall ui_i, ui_j: UI \cdot ui_i = ui_j \equiv \text{uid}_\text{NU}(ui_i) = \text{uid}_\text{NU}(ui_j) \)
5.2.5.1 Part Retrieval

• Given the unique identifier, $p_i$, of a part $p$, 
• but not the part itself, 
• and given the universe-of-discourse (uod) state $\sigma$, 
• we can retrieve part, $p$, as follows:

\[
\text{value} \\
\text{pre: } \exists p : P . \ p \in \sigma \land uid_P(p) = ui
\]
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 [32, 15].
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 types and
    * mereology observers (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 $E_{i0_1}, E_{i0_2}, ..., E_{i0_n}$ 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  \( iE_1 = iE_1 \mid iE_2 \mid \ldots \mid iE_m \)
62  \( ioE_1 = ioE_1 \mid ioE_2 \mid \ldots \mid ioE_n \)
63  \( oE_1 = oE_1 \mid oE_2 \mid \ldots \mid oE_o \)
60  \( MT = iE_1-set \times ioE_1-set \times oE_1-set \)

axiom
64  \( \forall (iset,ioset,oset):MT \cdot \)
64  \( \text{card} \ iset + \text{card} \ ioset + \text{card} \ oset = \text{card} \ \cup \{iset,ioset,oset\} \)
64  \( \cup \{iset,ioset,oset\} \subseteq \text{calc}_{all\_unique\_identifiers}(uod) \)
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. \textit{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] \( MT = \mathcal{M}(U_{i}, U_{j},..., U_{k}) \)

value
[m] \text{mereo\_P: P \rightarrow MT}

axiom [Well-formedness of Domain Mereologies ]
[a] \( A: A(MT) \)```

© Dines Bjørner, September 8, 2022
• 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 \( M(U_{l_i}, U_{l_j}, \ldots, U_{l_k}) \) is a type expression over unique identifiers.
  
  – Thus a part of type-name P might be given the mereology type name MP.

• \( A(MT) \) 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, 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.

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.

   - We presently omit treatment of road net and automobile aggregate mereologies.
   - For road net mereology we refer to Example 64, Item 130 on Slide 374.
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.

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.

— 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 $\forall (\text{uais}, \text{luis}) : H_{\text{Mer}} \cdot \text{luis} \subseteq l_{\text{uis}} \land \text{uais} = a_{\text{uis}}$
  66 $\forall (\text{uais}, \text{huis}) : L_{\text{Mer}} \cdot \text{uais} = a_{\text{uis}} \land \text{huis} \subseteq h_{\text{uis}} \land \text{card} \text{ huis} = 2$
  67 $\forall \text{ruis} : A_{\text{Mer}} \cdot \text{ruis} = r_{\text{uis}}$

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 $\forall h : H, l : L \cdot h \in hs \land l \in ls \Rightarrow$
  68 $\text{let } (\_, \text{luis}) = \text{mereo}_H(h), (\_, \text{huis}) = \text{mereo}_L(l)$
  69 $\text{in uid}_L(l) \in \text{luis} \equiv \text{uid}_H(h) \in \text{huis} \textbf{ end}$
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 \( \forall h_a, h_b : H, l : L \cdot \{h_a, h_b\} \subseteq hs \land l \in ls \Rightarrow \)

70 \( \textbf{let} (_,luis) = \text{meredo}_H(h), (_,huis) = \text{meredo}_L(l) \)

71 \( \textbf{in} \ uid_L(l) \in luis \equiv uid_H(h) \in huis \textbf{ end} \)
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 187,
  – “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 208,
  – 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\textsuperscript{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 210 is a \textit{fixed mereology}.

\textsuperscript{10} cf. Examples 21 on Slide 54, 29 on Slide 106, 30 on Slide 119, 32 on Slide 129, 35 on Slide 174, 36 on Slide 187, 38 on Slide 195, 39 on Slide 197, 40 on Slide 210 and 41 on Slide 213.
– 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 210 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 \rightarrow (UI-set \times UI-set)

axiom
80. \forall nu:NU .
80. let (uis_i,uis_o)=mereo_NU(nu) in
80. case (\text{card} uis_i,\text{card} usi_o) =
75. (is_LU(nu) \rightarrow (1,1),
76. is_PU(nu) \rightarrow (1,2) \lor (2,1),
77. is_RU(nu) \rightarrow (2,2),
78. is_SU(nu) \rightarrow (2,2), is_DU(nu) \rightarrow (2,2),
79. is_TU(nu) \rightarrow (1,0) \lor (0,1),
80. \_ \rightarrow \text{chaos}) end
80. \land uis_i \cap uis_o=\{
80. \land uid_NU(nu) \notin (uis_i \cup uis_o)
80. end
• Figure 5.1
  – illustrates the mereology of four rail units.

<table>
<thead>
<tr>
<th></th>
<th>Linear</th>
<th>Point</th>
<th>Rigid Crossing</th>
<th>Double Slip</th>
</tr>
</thead>
<tbody>
<tr>
<td>ua</td>
<td>____________</td>
<td>_____________</td>
<td>_____________</td>
<td>_____________</td>
</tr>
<tr>
<td>ui</td>
<td></td>
<td>_____________</td>
<td>_____________</td>
<td>_____________</td>
</tr>
<tr>
<td>ux</td>
<td></td>
<td></td>
<td>_____________</td>
<td>_____________</td>
</tr>
</tbody>
</table>

- 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’, $\phi$, has ‘happened’ to an endurant, $e_a$,
      · then some ‘commensurate thing’, $\psi$, 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 $\phi$ and $\psi$. 
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: $T = \mathcal{T}(...)$.  
- 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\rightarrow A_1, ..., attr A_m: E\rightarrow A_m

proof obligation [Disjointness of Attribute Types]

[p] 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) \neq is A_j(a) [i \neq i, i,j: [1..m]] end end”
• Let $A_1, \ldots, A_n$ 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, $A_1, \ldots, A_m$, i.e., $m < n$.
  
  – Across many domain models for “more-or-less the same” domain $m$ varies and the attributes, $A_1, \ldots, A_m$, selected for one model may differ from those, $A_1', \ldots, A_m'$, chosen for another model.
• The **type** definitions: $A_1, \ldots, A_m$, inform us that the domain analyser has decided to focus on the distinctly named $A_1, \ldots, A_m$ attributes.

• The **value** clauses
  
  – $\text{attr}_{A_1}: P \rightarrow A_1,$
  
  – $\ldots,$
  
  – $\text{attr}_{A_n}: P \rightarrow A_n$

  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 $\text{attr}_{A_i}: E \rightarrow A_i$ 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 [47] 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 [47, M.A.Jackson].
• Endurant attributes are
  – either constant, i.e., static,
  – or varying, i.e., dynamic

Attribute Category: 1.

• By a static attribute, a:A, \texttt{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*, $a:A$, $\text{is\_dynamic\_attribute}(a)$, 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, \( \text{is}_\text{reactive}_\text{attribute}(a) \), 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*, \( a:A, \text{is_active_attribute}(a) \), 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 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:

\[
\text{value} \\
\text{is_monitorable_only}: E \to \text{Bool} \\
\text{is_monitorable_only}(e) \equiv \text{is_inert}(e) \vee \text{is_reactive}(e) \vee \text{is_autonomous}(e)
\]
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 from link \(l_f\) to 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  $H\Sigma = (L_{UI} \times L_{UI})$-set
82  $H\Omega = H\Sigma$-set
83  $H_{\text{Traffic}} = (A_{UI} \mid B_{UI}) \rightarrow (\text{TIME} \times \text{VPos})^*$

axiom
81  $\forall h : H \cdot \text{obs}_{H\Sigma}(h) \in \text{obs}_{H\Omega}(h)$
83  $\forall ht : H_{\text{Traffic}}, ui : (A_{UI} \mid B_{UI}) \cdot ui \in \text{dom} ht \Rightarrow \text{time\_ordered}(ht(ui))$

value
81  $\text{attr}_{H\Sigma} : H \rightarrow H\Sigma$
82  $\text{attr}_{H\Omega} : H \rightarrow H\Omega$
83  $\text{attr}_{H_{\text{Traffic}}} : H \rightarrow H_{\text{Traffic}}$
83  $\text{time\_ordered} : (\text{TIME} \times \text{VPos})^* \rightarrow \text{Bool}$
83  $\text{time\_ordered}(tvpl) \equiv ...$

• In Item 83 we model the time-ordered sequence of traffic as a discrete sampling, i.e., $\rightarrow$, rather than as a continuous function, $\rightarrow$. 
Example 52. **Invariance of Road Net Traffic States:**

- We continue Example 51 on Slide 256.

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

**axiom**

84 $\forall h \in H \cdot \begin{cases} h \in hs \Rightarrow \\ \text{let } h\sigma = attr_{H\Sigma}(h) \in \\ \forall (l_{ui}i, l_{ui}i') : (L_{UI} \times L_{UI}) \cdot (l_{ui}i, l_{ui}i') \in h\sigma \Rightarrow \{l_{ui}i, l_{ui}i'\} \subseteq l_{ui}s \end{cases}$
• 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 from 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, \( h_{ui} \), of the road net’s hub identifiers.
type
85 $L\Sigma = H_{UI}$-set
86 $L\Omega = L\Sigma$-set
87 $L_{\text{Traffic}}$
87 $L_{\text{Traffic}} = (A_{UI}|B_{UI}) \overset{\rightarrow}{\rightarrow} (\mathbb{T}\times(H_{UI}\times\text{Frac}\times H_{UI}))^*$
87 $\text{Frac} = \text{Real, axiom } \text{frac:Fract} \cdot 0<\text{frac}<1$

class value
85 $\text{attr}_L\Sigma: L \rightarrow L\Sigma$
86 $\text{attr}_L\Omega: L \rightarrow L\Omega$
87 $\text{attr}_L_{\text{Traffic}}: : \rightarrow L_{\text{Traffic}}$

class axiom
85 $\forall l\sigma:\text{L}\Sigma \cdot \text{card } l\sigma = 2$
85 $\forall l:\text{L} \cdot \text{obs}_L\Sigma(l) \in \text{obs}_L\Omega(l)$
87 $\forall lt:L_{\text{Traffic}}, ui:(A_{UI}|B_{UI})\cdot ui \in \text{dom } ht \Rightarrow \text{time ordered}(ht(ui))$
87 $\forall l: L_{\text{Traffic}} \cdot l \in ls \Rightarrow \mathbf{let} l\sigma = \text{attr}_L\Sigma(l) \mathbf{in} \forall (h_{ui}, h_{ui}'): (H_{UI}\times K_{UI}) \cdot$
88 $\forall l: L \cdot l \in ls \Rightarrow \mathbf{let} l\sigma = \text{attr}_L\Sigma(l) \mathbf{in} \forall (h_{ui}, h_{ui}'): (H_{UI}\times K_{UI}) \cdot$
88 $\forall (h_{ui}, h_{ui}'): (H_{UI}\times K_{UI}) \cdot (h_{ui}, h_{ui}') \in l\sigma \Rightarrow \{h_{ui}, h'_{ui}\} \subseteq h_{ui}$'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. 257 and Sli. 261, 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 automobiles¹¹.

¹¹In 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 prompts:
  - `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.\(^{12}\)

\(^{12}\)\(\eta A\) is the type of all attribute types.
Observer Function Prompt 6. \textit{analyse\_attribute\_types}:

\begin{quote}
value
\begin{align*}
\text{analyse\_attribute\_type\_names: } & P \rightarrow \eta A\text{-set} \\
\text{analyse\_attribute\_type\_names(p) as } & \{\eta A_1, \eta A, \ldots, \eta A_m\}
\end{align*}
\end{quote}
Observer Function Prompt 7. *sta_attr_types*:

value

\[ \text{sta_attr_types}: P \rightarrow \eta A \times \eta A \times \ldots \times \eta A \]

**sta_attr_types(p) as** \((\eta A_1, \eta A_2, \ldots, \eta A_n)\)

**where:**

\[ \{\eta A_1, \eta A_2, \ldots, \eta A_n\} \subseteq \text{analyse_attribute_type_names}(p) \]

\[ \land \text{let } anms = \text{analyse_attribute_type_names}(p) \]

\[ \forall \text{ anm:} \eta A \cdot \text{ anm } \in \text{anms} \setminus \{\eta A_1, \eta A_2, \ldots, \eta A_n\} \]

\[ \Rightarrow \sim \text{ is_static_attribute}\{\text{anm}\} \]

\[ \land \forall \text{ anm:} \eta A \cdot \text{ anm } \in \{\eta A_1, \eta A_2, \ldots, \eta A_n\} \]

\[ \Rightarrow \text{ is_static_attribute}\{\text{anm}\} \text{ end} \]
Observer Function Prompt 8. \textit{mon_attr_types}:

\begin{verbatim}
value
\textit{mon_attr_types}: P → ηA×ηA×...×ηA
\textit{mon_attr_types}(p) as (ηA1,ηA2,...,ηAn)
   \textbf{where:} \{ηA1,ηA2,...,ηAn\} ⊆ \text{analyse_attribute_type_names}(p)
   \land \text{let anms = analyse_attribute_type_names}(p)
   \land \forall anm:ηA • anm ∈ anms \setminus \{ηA1,ηA2,...,ηAn\}
   ⇒ \sim \text{is_monitorable_attribute\{anm\}}
   \land \forall anm:ηA • anm ∈ \{ηA1,ηA2,...,ηAn\}
   ⇒ \text{is_monitorable_attribute\{anm\}} \textbf{end}
\end{verbatim}
Observer Function Prompt 9. *pro_attr_types*:

**value**

\[
\text{pro_attr_types: } P \rightarrow \eta\Delta \times \eta\Delta \times \ldots \times \eta\Delta
\]

\[
\text{pro_attr_types(p) as } (\eta A_1, \eta A_2, \ldots, \eta A_n)
\]

**where:** \{\eta A_1, \eta A_2, \ldots, \eta A_n\} ⊆ analyse_attribute_type_names(p)

\[
\land \text{ let } anms = \text{analyse_attribute_type_names(p)}
\]

\[
\forall \text{ anm:}\eta\Delta \cdot \text{anm} \in \text{anms} \setminus \{\eta A_1, \eta A_2, \ldots, \eta A_n\}
\]

\[
\Rightarrow \sim \text{is_monitorable_attribute}{\text{anm}}
\]

\[
\land \forall \text{ anm:}\eta\Delta \cdot \text{anm} \in \{\eta A_1, \eta A_2, \ldots, \eta A_n\}
\]

\[
\Rightarrow \text{is_monitorable_attribute}{\text{anm}} \text{ end}
\]
• 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 \((\eta A_1, \eta A_2, \ldots, \eta A_n)\) be a grouping of attribute types for part \(p\) (or fluid \(f\)).
• Then \((\text{attr}_{A_1}(p), \text{attr}_{A_2}(p), \ldots, \text{attr}_{A_n}(p))\)
• (respectively \(f\))
• yields \((a_1, a_2, \ldots, a_n)\), the grouping of values for these attribute types.
• We can “formalise” this conversion:

\[
\text{value types_to_values: } \eta A_1 \times \eta A_2 \times \ldots \times \eta A_n \rightarrow A_1 \times A_2 \times \ldots \times A_n
\]
5.4.2.7 Calculating Attribute Names

• The meta-linguistic, i.e., “outside” RSL proper, name for attribute type names is introduced here as $\eta A$.

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.

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 $e$.

¹³By 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 monitorable, i.e., monitorable-only and biddable, attributes,
- sets, PT, of names of programmable, i.e., fully controllable attributes.

type

91  \( ST = \eta A\text{-set} \)
92  \( MA = \eta A\text{-set} \)
93  \( PT = \eta A\text{-set} \)

value

91  \( \text{stat_attr_types}: E \rightarrow ST \)
92  \( \text{moni_attr_types}: E \rightarrow MA \)
93  \( \text{prgr_attr_types}: E \rightarrow PT \)
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, $\text{stat\_attr\_vals}$;
96. given endurant $e$ we can *meta-linguistically* calculate its monitorable-only attribute values, $\text{moni\_attr\_vals}$; and
97. given endurant $e$ we can *meta-linguistically* calculate its programmable attribute values, $\text{prgr\_attr\_vals}$.

The type names $\text{sa1, \ldots, pap}$ refer to the types denoted by the corresponding types name $\text{nsa1, \ldots, npap}$.
value
95 stat_attr_vals: \( E \rightarrow \text{SA1} \times \text{SA2} \times \ldots \times \text{SAs} \)
95 \( \text{stat_attr_vals}(e) \equiv \)
95 \begin{align*}
&\text{let } \{ \text{nsa1, nsa2, \ldots, nsas} \} = \text{stat_attr_types}(e) \text{ in} \\
&(\text{attr}_{\text{sa1}}(e), \text{attr}_{\text{sa2}}(e), \ldots, \text{attr}_{\text{sas}}(e)) \text{ end}
\end{align*}

96 moni_attr_vals: \( E \rightarrow \text{MA1} \times \text{MA2} \times \ldots \times \text{MAm} \)
96 \( \text{moni_attr_vals}(e) \equiv \)
96 \begin{align*}
&\text{let } \{ \text{nma1, nma2, \ldots, nmam} \} = \text{moni_attr_types}(e) \text{ in} \\
&(\text{attr}_{\text{ma1}}(e), \text{attr}_{\text{ma2}}(e), \ldots, \text{attr}_{\text{mam}}(e)) \text{ end}
\end{align*}
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 155.
    * 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 $\sigma$. 
• Given a part, \( p \), with attribute \( A \),
  – the simple operation \( \text{attr}_A(p) \)
  – thus yields the value of attribute \( A \)
  – for that part.

• But what if, what we have is just
  – the global state \( \sigma \), of the set of all monitorable parts of a given
    universe-of-discourse, \( \text{uod} \),
  – the unique identifier, \( \text{uid}_P(p) \), of a part of \( \sigma \), and
  – the name, \( \eta_A \), of an attribute of \( p \)?
    * Then how do we
      · ascertain the attribute value for \( A \) of \( p \),
      · and, for \( \text{biddable} \) attributes \( A \),
      · “update” \( p \), in \( \sigma \), 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 $\eta_A$ be the name of attribute $A$.

99. Evaluation of the [current] attribute $A$ value of $p$ is defined by function $\text{read}_A \_\text{from}_P$ – $\text{retr}_\text{part}(\pi)$ is defined in Sect. 5.2.5.1 on Slide 198.

value

98. $\pi:PI$, $a:A$, $\eta_A:\eta \, T$

99. $\text{read}_A \_\text{from}_P: PI \times T \rightarrow \text{read} \, \sigma$

99. $\text{read}_A(\pi, \eta_A) \equiv \text{attr}_A(\text{retr}_\text{part}(\pi))$
5.4.3.2 Update of Biddable Attributes

100. The update of a monitorable attribute $A$, with attribute name $\eta A$ of part $p$, identified by $p_i$, to a new value \texttt{writes} to the global part state $\sigma$.

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 $\sigma$. 
value

98. \[ \sigma, a:A, \pi:Pl, \eta A: \eta T \]

100. update\_P\_with\_A: Pl \times A \times \eta T \to \textbf{write } \sigma \\
100. \text{update}\_P\_with\_A(\pi,a,\eta A) \equiv \\
101. \textbf{let} \ p = \text{retr\_part}(\pi) \ \textbf{in} \\
102. \textbf{let} \ p:P. \\
102a. \ \text{uid\_P}(p)=\pi \\
102b. \ \land \ \text{mereo\_P}(p)=\text{mereo\_P}(p) \\
102c. \ \land \ \forall \ \eta A' \ \textbf{in} \ \text{analyse\_attribute\_type\_names}(p) \ \backslash \ \{\eta A\} \\
102c. \ \Rightarrow \ \text{attr\_A}(p)=\text{attr\_A}(p) \\
102d. \ \land \ \text{attr\_A}(p)=a \ \textbf{in} \\
103. \ \sigma := \sigma \ \backslash \ \{p\} \ \cup \ \{p\} \\
100. \ \textbf{end } \textbf{end}
5.5 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 Philosophy [61, 62, 63, 64].
5.5.1 Issues Leading Up to Intentionality

5.5.1.1 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.”
5.5.1.2  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.”
5.5.1.3 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.”
5.5.1.4 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.”
5.5.1.5 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.”
5.5.1.6 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."
5.5.1.7 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.

  • • •
5.5.2 Intentionality

• *Intentionality* as
  – a philosophical concept
  – is defined by the Stanford Encyclopedia of Philosophy\(^\text{14}\) as
    \* “*the power of minds to be about,*
    \* *to represent, or to stand for,*
    \* *things, properties and states of affairs.*”

---

5.5.2.1 Intentional Pull

• Two or more artefactual parts
  – of different sorts, but with overlapping sets of intents
  – may exert 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*, $\iota$.
  – *Manifestations* of these, their common intent
  – must somehow be *subject to constraints*,
  – and these must be *expressed predicatively*. 
Example 54. **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\textsuperscript{15} 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.

\textsuperscript{15}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].
5.5.2.2 **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*
5.5.2.3 Intentionalities

Observer Function Prompt 10. \textit{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 \textit{domain analysis prompt}:

  \textit{analyse\_intentionality} directs the domain analyser to observe a set of intents.

  \textbf{value} \textit{analyse\_intentionality}(e) \equiv \{i_1,i_2,...,i_n\}\subseteq\text{Intent}
Example 55. **Intentional Pull: Road Transport:**

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

104. 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.

<table>
<thead>
<tr>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>104a. LHist = A</td>
<td> </td>
</tr>
<tr>
<td>104b. HHist = A</td>
<td></td>
</tr>
<tr>
<td>104c. AHist = (L</td>
<td>H)</td>
</tr>
</tbody>
</table>
5.5.2.4 Wellformedness of Event Histories

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

105. Each $\tau_\ell : \text{TIME}^*$ is an indefinite list.
   We have not expressed any criteria
   for the recording of events: *all the time, continuously* ! (?)

106. Each list of times, $\tau_\ell : \text{TIME}^*$, is here to be in decreasing, *continuous* order of times.

107. 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.

108. If an automobile leaves a link (a hub), at time $\tau$, then it may enter a hub (resp. a link)
   and then that must be at time $\tau'$
   where $\tau'$ is some infinitesimal, sampling time interval, quantity larger than $\tau$.
   Again we refrain here from speculating on the issue of sampling!

109. 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!
5.5.2.5 Formulation of an Intentional Pull

110. 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.

111. 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

110a. \( \forall a:A \cdot a \in as \Rightarrow \)

110a. `let ahist = attr_AHist(a)` `in`

110a. \( \forall \, \text{ui}:(L||H) \cdot \text{ui} \in \text{dom} \, \text{ahist} \Rightarrow \)

110b. \( \forall \, \tau:TIME \cdot \tau \in \text{elems} \, \text{ahist}(\text{ui}) \Rightarrow \)

110b. `let hist = is_LL(\text{ui}) \rightarrow attr_LHist(retr_L(\text{ui}))(\sigma),`

110b. `_ \rightarrow attr_HHist(retr_H(\text{ui}))(\sigma) \, \text{in}`

110b. \( \tau \in \text{elems} \, \text{hist}(\text{uid}_A(a)) \, \text{end \ end} \)

111. \( \wedge \)

111c. \( \forall \, u:(L\,|\,H) \cdot u \in ls \cup hs \Rightarrow \)

111c. `let uhist = attr(L\,|\,H)Hist(u)` `in`

111d. \( \forall a_i:Ai \cdot a_i \in \text{dom} \, \text{uhist} \Rightarrow \)

111d. \( \forall \, \tau:TIME \cdot \tau \in \text{elems} \, \text{uhist}(a_i) \Rightarrow \)

111d. `let ahist = attr_AHist(retr_A(a_i))(\sigma)` `in`

111d. \( \tau \in \text{elems} \, \text{uhist}(a_i) \, \text{end \ end} \)
• 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 56. 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 56 on the facing 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.
5.5.3 **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*. 
5.5.4 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\(^{16}\) of
  – the International System of Units:
• Attribute values usually enter into differential equations and integrals,
• that is, classical calculus.

\(^{16}\)Basic units are meter, kilogram, second, Ampere, Kelvin, mole, and candela. Some derived units are: Newton: \(kg \times m \times s^{-2}\), Weber: \(kg \times m^2 \times s^{-2} \times A^{-1}\), 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 57. **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.
5.5.5 Galois Connections

- Galois Theory was first developed by Évariste Galois [1811-1832] around 1830\textsuperscript{17}.
- Galois theory emphasizes a notion of Galois connections.
- We refer to standard textbooks on Galois Theory, e.g., [67, 2009].

\textsuperscript{17}en.wikipedia.org/wiki/Galois_theory
5.5.5.1 Galois Theory: An Ultra-brief Characterisation

- To us, an essence of Galois connections can be illustrated as follows:
  - Let us observe\(^{18}\) properties of a number of endurants, say in the form of attribute types.
  - Let the function \(F\) map sets of entities to the set of common attributes.
  - Let the function \(G\) map sets of attributes to sets of entities that all have these attributes.
  - \((F, G)\) 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.
  * \((F, G)\) is monotonously decreasing.

\(^{18}\)The following is an edited version of an explanation kindly provided by Asger Eir, e-mail, June 5, 2020 [36, 37, 27].
Example 58. LEGO Blocks:

- We 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 $B = \{\text{attr\_Color}(e) = \text{red}, \text{attr\_Form}(e) = \text{square}\}$.
  - We now add all the blue square blocks.
  - And obtain $A'$.

- Now the common properties are their \text{squareness}:
  - $\mathcal{F}(A')$ is $B' = \{\text{attr\_Form}(e) = \text{square}\}$.

- 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. \hfill \blacksquare
Example 59. **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\(^\text{20}\)
5.5.5.2 **Galois Connections and Intentionality – A Possible Research Topic?**

• We have a hunch\(^{21}\)!
  - 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.

\(^{21}\)Hunch: a feeling or guess based on intuition rather than fact.
5.5.6 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” endurants
  – and reflects on whether they stand in an intentional pull relation to one another.
• We refer to Sects. 5.5.2.2 – 5.5.2.3.

---

22By “independent” we shall here mean that these endurants are not ‘derived’ from one-another!
5.6 A Domain Discovery Procedure, II

• We continue from Sect. 4.8.

5.6.1 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 158).

• 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() ≡ ...

© Dines Bjørner, September 8, 2022
value

discover_internal_qualities: \textbf{Unit} \rightarrow \textbf{Unit}

discover_internal_qualities() \equiv

discover_uids();

\textbf{axiom} [ all parts have unique identifiers ]

discover_mereologies();

\textbf{axiom} [ all unique identifiers are mentioned in sum total of ]

[ all mereologies and no isolated proper sets of parts ]

discover_attributes();

\textbf{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.
5.6.2 A Suggested Analysis & Description Approach, II

• Figure 4.3 on Slide 129 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 156 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 127) 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 129 and Example 29 on Slide 106, thus gives rise to many different analysis & description traversals.

– Figure 5.3 on the following slide illustrates one such order.
– 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.
## Summary

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Introduced</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analysis Predicates</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>is_manifest</td>
<td>page 173</td>
</tr>
<tr>
<td>17</td>
<td>is_structure</td>
<td>page 173</td>
</tr>
<tr>
<td><strong>Attribute Analysis Predicates</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>is_static_attribute</td>
<td>page 239</td>
</tr>
<tr>
<td>2</td>
<td>is_dynamic_attribute</td>
<td>page 241</td>
</tr>
<tr>
<td>3</td>
<td>is_inert_attribute</td>
<td>page 242</td>
</tr>
<tr>
<td>4</td>
<td>is_reactive_attribute</td>
<td>page 244</td>
</tr>
<tr>
<td>5</td>
<td>is_active_attribute</td>
<td>page 246</td>
</tr>
<tr>
<td>6</td>
<td>is_autonomous_attribute</td>
<td>page 247</td>
</tr>
<tr>
<td>7</td>
<td>is_biddable_attribute</td>
<td>page 249</td>
</tr>
<tr>
<td>8</td>
<td>is_programmable_attribute</td>
<td>page 251</td>
</tr>
<tr>
<td>9</td>
<td>is_monitorable_attribute</td>
<td>page 255</td>
</tr>
<tr>
<td><strong>Analysis functions</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>analyse_attribute_types</td>
<td>page 269</td>
</tr>
<tr>
<td>7</td>
<td>sta_attr_types</td>
<td>page 270</td>
</tr>
<tr>
<td>8</td>
<td>mon_attr_types</td>
<td>page 271</td>
</tr>
<tr>
<td>9</td>
<td>pro_attr_types</td>
<td>page 272</td>
</tr>
<tr>
<td><strong>Retrieval, Read and Write functions</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>read_A from P</td>
<td>page 283</td>
</tr>
<tr>
<td>100</td>
<td>update_P with A</td>
<td>page 284</td>
</tr>
<tr>
<td><strong>Description Functions</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>describe_unique_identifier</td>
<td>page 183</td>
</tr>
<tr>
<td>6</td>
<td>describe_mereology</td>
<td>page 207</td>
</tr>
<tr>
<td>7</td>
<td>describe_attributes</td>
<td>page 233</td>
</tr>
<tr>
<td><strong>Domain Discovery</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>discover_uids</td>
<td>page 325</td>
</tr>
<tr>
<td>6</td>
<td>discover_mereologies</td>
<td>page 325</td>
</tr>
<tr>
<td>7</td>
<td>discover_attributes</td>
<td>page 325</td>
</tr>
<tr>
<td>8</td>
<td>discover_internal_qualities</td>
<td>page 325</td>
</tr>
</tbody>
</table>

© Dines Bjørner, September 8, 2022
Lecture 6: Perdurants, I
CHAPTER 6. Perdurants

• This chapter is a rather “drastic” reformulation and simplification of [24, Chapter 7, i.e., pages 159–196].
  – Besides, Sect. 6.5 is new.

• In this chapter we transcendentally “morph” manifest
  – parts into behaviours, that is:
  – endurants into perdurants.

• We analyse that notion and its constituent notions of
  – actors,
  – channels and communication,
  – actions and
  – behaviours.
• We shall investigate the, as we shall call them, perdurants of domains.
• That is state and time-evolving domain phenomena.
• The outcome of this chapter is that the student
  – will be able to model the perdurants of domains.
  – Not just for a particular domain instance,
  – but a possibly infinite set of domain instances\(^1\).

---

\(^1\)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.
6.1 Part Behaviours – An Analysis

6.1.1 Behaviour Definition Analysis

- Parts co-exist;
  - they do so endurantly as well as perdurantly:
    - endure and perdure.

- Part perdurants, i.e., behaviours, interact with their surroundings, that is, with other behaviours.

- This is true for both natural and man-made parts.

- The present domain modelling method is mainly focused on man-made parts, that is artefacts.

- So our next analysis will take its clues from artefactual parts.

- We can, roughly, analyse part behaviours into three kinds.
• **Proactive Behaviours:** Behaviour $B_i$ offers to synchronise and communicate values – *internal non-deterministically* with either of a definite number of distinct part sort behaviours $B_a, B_b, ..., B_c$:

$$B(i)(\text{args}) \equiv 
\begin{align*}
(... \text{ch[\{i,a\}]} ! a_{\text{val}} ; ... ; B(i)(\text{args}'))
\land (...
\text{ch[\{i,b\}]} ! b_{\text{val}} ; ... ; B(i)(\text{args}''))
\land ...
\land (...
\text{ch[\{i,c\}]} ! c_{\text{val}} ; ... ; B(i)(\text{args}''''))
\end{align*}
$$

The tail-recursive invocation of $B_i$ indicates a possible “update” of behaviour $B_i$ arguments. More on this later.
• **Responsive Behaviours:** Behaviour $B_i$ *external* non-deterministically expresses willingness to synchronisation with and accept values from either of a definite number of distinct part sort behaviours $B_a, B_b, ..., B_c$:

$$B(i)(\text{args}) \equiv \begin{array}{c}
\ldots \text{let } av = ch[\{i,a\}] ? \text{ in } \ldots B(i)(\text{args'}) \text{ end} \\
\ll(\ldots \text{let } bv = ch[\{i,b\}] ? \text{ in } \ldots ; B(i)(\text{args''}) \text{ end} \\
\ll(\ldots \text{let } cv = ch[\{i,c\}] ? \text{ in } \ldots ; B(i)(\text{args''}) \text{ end}
\end{array}$$
• **Mixed Behaviours:** Or behaviours, more generally, “are” an internal non-deterministic “mix” of the above:

\[
\begin{align*}
\text{B}(i)(\text{args}) \equiv & \quad ((\ldots \text{ch}[\{i,a\}] \!\! \text{! a\_val} \ldots \text{; B}(i)(\text{args}')) \\
& \quad (\ldots \text{ch}[\{i,b\}] \!\! \text{! b\_val} \ldots \text{; B}(i)(\text{args}'')) \\
& \quad (\ldots \text{ch}[\{i,c\}] \!\! \text{! c\_val} \ldots \text{; B}(i)(\text{args}''))) \\
& \quad ((\ldots \text{let av} = \text{ch}[\{i,a\}] \? \text{in} \ldots \text{B}(i)(\text{args}’) \text{end}) \\
& \quad (\ldots \text{let bv} = \text{ch}[\{i,b\}] \? \text{in} \ldots \text{; B}(i)(\text{args}'') \text{end}) \\
& \quad (\ldots \text{let cv} = \text{ch}[\{i,c\}] \? \text{in} \ldots \text{; B}(i)(\text{args}'’’) \text{end})
\end{align*}
\]
• The “bodies” of the $B_i$ behaviour definitions, i.e., “…” , may contain interactions with [yet other] behaviours. Schematically for example:

\[
\text{let } yv = \text{ch}[\{i,y\}] \ ? \ \text{in} \ldots \text{end}
\]

\[
\text{let } zv = \emptyset \{ \text{ch}[\{i,z\}] \ ? \ | z:\{z1,z2,\ldots,zm\} \} \ \text{in} \ldots \text{end}
\]

Etcetera. The full force of CSP with RSL is at play!
6.1.2 Channel Analysis

• This is the first of two treatments of the concept of channels; the present treatment is informal, motivational, the second treatment, Sect. 6.2 (right next!), is more formal.

• The CSP concept of channel is to be our way of expressing the “medium” in which behaviours interact.
  – Channels is thus an abstract concept.
  – Please do not think of it as a physical, an IT (information technology) device.
  – As an abstract concept it is defined in terms of, roughly, the laws, the semantics, of CSP [44].
  – We write ‘roughly’ since the CSP we are speaking of, is “embedded” in RSL.
6.2 Domain Channel Description

- We simplify the general treatment of channel declarations.
  - Basically all we can say, for any domain,
  - is that any two distinct part behaviours
  - may need to communicate.
  - Therefore we declare a vector of channels
  - indexed by sets of two distinct part identifiers.

\[
\text{value} \\
\text{discover\_channels: } \text{Unit} \rightarrow \text{Unit} \\
\text{discover\_channels}() \equiv \\
\quad \text{``channel } \{ \text{ch}[\{ij,ik\}] \mid ij,ik:Ul \cdot \{ij,ik\} \subseteq \text{uid}_o \land ij \neq ik \} \text{ M }``
\]

- Initially we shall leave the type of messages over channels further undefined.
- As we, laboriously, work through the definition of behaviours, we shall be able to make M precise.
6.3 Behaviour Definition Description

• Behaviours have to be described.
  – Behaviour definitions are in the form of function definitions and are here expressed in RSL relying, very much, on its CSP component.
  – Behaviour definitions describe the type of the arguments the function, i.e., the behaviour, for which it is defined, that is, which kind of values it accepts.
  – Behaviour definitions further describe
• Thus there are two elements to a behaviour definition:
  – the behaviour signature and
  – the behaviour body definitions.
6.3.1 Behaviour Signatures

6.3.1.1 General

- Function, F, signatures consists of two textual elements:
  - the function name and
  - the function type:
    
    **value** F: A → B, or F: a:A → B
    
  - where A and B are the types of
    * function ("input") arguments, respectively
    * function ("output") values for such arguments.
  - The first form F: A → B is what is normally referred to as the form for function signatures.
6.3.1.2 Domain Behaviour Signatures

• A schematic form of part \((p)\) behaviour signatures is:

\[ b: \text{bi:Bl} \rightarrow \text{me:Mer} \rightarrow \text{svl:StaV^*} \rightarrow \text{mvl:MonV^*} \rightarrow \text{prgl:PrgV^*} \text{ channels Unit} \]

• We shall motivate the general form of part behaviour, \(B\), signatures, “step-by-step”: 
α. the [chosen] name of part \( p \) behaviours.

β. \( U \to V \to \ldots \to W \to Z \): The function signature is expressed in the Schönfinkel/Curry style – corresponding to the invocation form \( F(u)(v)\ldots(w) \)

γ. \( bi:Bl \): a general value and the type of part \( p \) unique identifier

δ. \( me:Mer \): a general value and the type of part \( p \) mereology

ε. \( svl:StaV^* \): a general (possibly empty) list of values and types of part \( p \)’s (possibly empty) list of static attributes

ζ. \( mvl:MonV^* \): a general list of names of types of part \( p \)’s (possibly empty) list of monitorable attributes

η. \( prgl:PrgV^* \): a general list of values and types of part \( p \)’s (possibly empty) list of programmable attributes

θ. channels: are usually of the form: \( \{ch[\{i,j\}] | (i,j) \in I(\text{me})\} \) and express the subset of channels over which behaviour \( B_i \)s interact with other behaviours

ι. \( Unit \): designates the single value ()

---

2Moses Schönfinkel (1888–1942) was a Russian logician and mathematician accredited with having invented combinatory logic [https://en.wikipedia.org/wiki/Moses_Schönfinkel]. Haskell B. Curry (1900–1982) was an American mathematician and logician known for his work in combinatory logic [https://en.wikipedia.org/wiki/Haskell_Curry]
In detail:

α. **Behaviour name:** In each domain description there are many sorts, B, of parts. For each sort there is a generic behaviour, whose name, here b, is chosen to suitably reflect B.

β. **Currying** is here used in the pragmatic sense of grouping “same kind of arguments”, i.e., separating these from one-another, by means of the →s.

γ. The **unique identifier** of part sort B is here chosen to be Bl. Its value is a constant.

δ. The **mereology** is a usually constant. For same part sorts it may be a variable.
Example 60. **Variable Mereologies:**

- For a road transport system where we focus on the transport the mereology is a constant.

- For a road net where we focus on the development of the road net: building new roads: inserting and removing hubs and links, the mereology is a variable.

- Similar remarks apply to canal systems [14], pipeline systems [14], container terminals [20], assembly line systems [21], etc.
Static attribute values are constants. The use of static attribute values in behaviour body definitions is expressed by an identifier of the stvl list of identifiers.

Monitorable attribute values are generally, ascertainable, i.e., readable, cf. Sect. 5.4.3.1 on Slide 283. Some are biddable, can be changed by a, or the behaviour, cf. Sect. 5.4.3.2 on Slide 284, but there is no guarantee, as for programmable attributes, that they remain fixed.

– The use of a[ny] monitorable attribute value in behaviour body definitions is expressed by a read_A_from_P(mv,bi) where mv is an identifier of the mvl list of identifiers and bi is the unique part identifier of the behaviour definition in which the read occurs.

– The update of a biddable attribute value in behaviour body definitions is expressed by a update_P_with_A(bi,mv,a).
η. **Programmable attribute values** are just that. They vary as specified, i.e., “programmed”, by the behaviour body definition. Tail-recursive invocations of behaviour $B_i$ “replace” relevant programmable attribute argument list elements with “new” values.

θ. **channels:** $I$(me) expresses a set of unique part identifiers different from $b_i$, hence of behaviours, with which behaviour $b(i)$ interacts.

ι. The **Unit** of the behaviour signature is a short-hand for the behaviour either **reading** the value of a monitorable attribute, hence global state $\sigma$, or performing a **write**, i.e., an **update**, on $\sigma$. 

© Dines Bjørner, September 8, 2022
6.3.1.3 Action Signatures

• Actions come in any forms:

112. Some take no arguments, say action_a(), but read the global state component \( \sigma \), and

113. others also take no arguments, say action_b(), but update the global state component \( \sigma \).

114. Some take an argument, say, action_c(c), but do not “touch” a global state component,

115. while others both take an argument and deliver a value, say action_d(d) and also do not “touch” a global state component.

116. Et cetera!
type A, B, C, D, ...

value

112. action_a: Unit → read σ A
113. action_b: Unit → write σ B
114. action_c: C → Unit
115. action_d: D → E Unit
116. ...

• An example of 114 are the CSP output: ch[...]! c, and
• an example of 115 are the CSP input: let e = ch[...]? in ... end.
6.3.2  Behaviour Invocation

• The general form of behaviour invocation is shown below.
  – The invocation follows the “Currying” of the behaviour type signature.
  – [Normally one would write all this on one line: b(i)(m)(s)(m)(p) ≡.]

  behaviour_name
  (unique_identifier)
  (mereology)
  (static_values)
  (monitorable_attribute_names)
  (programmable_variables) ≡

  ... body ...

• When first “invoked”: 
value
discoverBehaviourSignature: P \rightarrow \text{RSL-Text}
discoverBehaviourSignature(p) \equiv
"behaviour_name:
  Uld \rightarrow Mereo \rightarrow StaVL \rightarrow MonVL \rightarrow ProVL \rightarrow \text{channels Unit}
  behaviour_name
  (uid_B(p))
  (mereo_B(p))
  (types_to_values(static_attribute_types(p)))
  (mon_attribute_types(p))
  (types_to_values(programmable_attribute_types(p))) \equiv ")
pre: is_B(p) \land is_manifest(p)

discoverBehaviourSignatures: \text{Unit} \rightarrow \text{RSL-Text}
discoverBehaviourSignatures() \equiv
\{ \text{discoverBehaviourSignature}(p) \mid p \in \sigma \land \text{is_manifest}(p) \}
6.3.3 **Behaviour Definition Bodies**

• We remind the student of Sect. 6.1.1 on Slide 333.

• The general, “mixed”, form of behaviour definitions was given as:

\[
B(i)(\text{args}) \equiv \\
( ( ... ) \\
\| ( ... \ ch[i,b] ! b_{\text{val}} ; ... ; B(i)(\text{args''}) ) \\
\| ( ... ) ) \\
\| ( ( ... ) \\
\| ( ... ) \ \text{let} \ bv = ch[i,b] ? \text{in} ... ; B(i)(\text{args'') end } ) \\
\| ( ... ) )
\]
• We can express the same
  – by separating the alternatives
  – into invocations of separately defined behaviors.

\[ \text{B}(i)(\text{args}) \equiv \]
\[ ( \ldots
  \parallel \text{Bin}_j(i)(\text{args})
  \parallel \ldots )
  \parallel ( \ldots
    \parallel \text{Bxn}_k(i)(\text{args})
    \parallel \ldots ) \]

• where
  – the internal don-deterministically invoked behaviors \( \text{Bin}_j(i)(\text{args}) \)
  – the external don-deterministically invoked behaviors \( \text{Bin}_k(i)(\text{args}) \)

• are then separately defined:

\[ \text{Bin}_j(i)(\text{args}) \equiv ( \ldots \text{Bin}_j(i)(\text{args}^\prime) ) \]
\[ \text{Bxn}_k(i)(\text{args}) \equiv ( \ldots \text{Bxn}_k(i)(\text{args}^\prime\prime) ) \]
6.3.4 Discover Behaviour Definition Bodies

• In other words,
  – for current lack of a more definitive methodology
  – for “discovering” the bodies of behaviour definitions
  – we resort to “…”!

\textbf{value}

\begin{align*}
\text{discover\_behaviour\_definition}: \ & P \rightarrow \text{RSL-Text} \\
\text{discover\_behaviour\_definition}(p) \equiv & \ldots
\end{align*}

\begin{align*}
\text{discover\_behaviour\_definitions}: \ & \text{Unit} \rightarrow \text{RSL-Text} \\
\text{discover\_behaviour\_definitions}() \equiv & \\
& \{ \text{discover\_behaviour\_definition}(p) \mid p \in \sigma \land \text{is\_manifest}(p) \} 
\end{align*}
Example 61. **Automobile Behaviour:**

**Signatures**

117. automobile:

(a) there is the usual “triplet” of arguments: unique identifier, mereology and static attributes;

(b) then there are two programmable attributes: the automobile position (cf. Item 90 on Slide 263), and the automobile history (cf. Item 104c on Slide 303);

(c) and finally there are the input/output channel references allowing communication between the automobile and the hub and link behaviours.

118. Similar for

(a) link and

(b) hub behaviours.
• We omit the modelling of monitorable attributes (...).

**value**

117a,117a automobile: ai:AI $\rightarrow$ (__,uis):AM $\rightarrow$ ...

117b $\rightarrow$ (apos:APos $\times$ ahist:AHist)

117c $\textbf{in out} \{\text{ch[\{ai,ui\}]}|ai:AI,ui:(HI|LI) \cdot ai\in ais \land ui \in uis\} \textbf{Unit}$

118a link: li:LI $\rightarrow$ (his,ais):LM $\rightarrow$ LΩ $\rightarrow$ ...

118a $\rightarrow$ (LΣ×L_Hist)

118a $\textbf{in out} \{\text{ch[\{li,ui\}]}|li:LI,ui:(AI|HI)-set \cdot ai\in ais \land li \in lis\cup his\} \textbf{Unit}$

118b hub: hi:HI $\rightarrow$ (__,ais):HM $\rightarrow$ HΩ $\rightarrow$ ...

118b $\rightarrow$ (HΣ×H_Host)

118b $\textbf{in out} \{\text{ch[\{ai,ui\}]}|hi:HI,ai:AI \cdot ai\in ais \land hi \in uis\} \textbf{Unit}$
Definitions: Automobile at a Hub

119. We abstract automobile behaviour at a Hub (hi).
   (a) Either the automobile remains in the hub,
   (b) or, internally non-deterministically,
   (c) leaves the hub entering a link,
   (d) or, internally non-deterministically,
   (e) stops.

\[
\text{automobile}(ai)(aai,uis)(...)\text{atH}(fli,hi,tli),ahist) \equiv \\
\text{automobile}\_\text{remains\_in\_hub}(ai)(aai,uis)(...)\text{atH}(fli,hi,tli),ahist) \cap \\
\text{automobile}\_\text{leaving\_hub}(ai)(aai,uis)(...)\text{atH}(fli,hi,tli),ahist) \cap \\
\text{automobile}\_\text{stop}(ai)(aai,uis)(...)\text{atH}(fli,hi,tli),ahist)
\]
120. [119a] The automobile remains in the hub:
   (a) the automobile remains at that hub, “idling”,
   (b) informing (“first”) the hub behaviour.

\[
\begin{align*}
\text{120 automobile\_remains\_in\_hub}(ai)(aai,uis)(...)(apos:atH(fli,hi,tli),ahist) & \equiv \\
\text{120 let } \tau = \text{record\_TIME()} \text{ in } \\
\text{120b ch}[ai,hi]!\tau; \\
\text{120a automobile}(ai)(aai,uis)(...)(apos,upd\_hist(\tau,hi)(ahist)) \\
\text{120 end}
\end{align*}
\]

\[
\begin{align*}
\text{120a upd\_hist: (TIME} \times I) & \to \ (AHist|LHist|HHist) \to \ (AHist|LHist|HHist) \\
\text{120a upd\_hist}(\tau,i)(hist) & \equiv \text{hist} \uplus [i \mapsto \langle \tau \rangle \text{hist}(i)]
\end{align*}
\]
121. [119c] The automobile leaves the hub entering a link:

(a) tli, whose “next” hub, identified by thi, is obtained from the mereology of the link identified by tli;

(b) informs the hub it is leaving and the link it is entering,

(c) “whereupon” the vehicle resumes (i.e., “while at the same time” resuming) the vehicle behaviour positioned at the very beginning (0) of that link.

121  automobile_leaving_hub(ai)(aai,uis)(...)(apos:atH(fli,hi,tli),ahist) ≡
121a  (let ({fhi,thi},ais) = mereo_Ł(retro_Ł(tli)(σ)) in assert: fhi=hi
121b  ( ch[ai,hi]!τ ∥ ch[ai,tli]!τ );
121c  automobile(ai)(aai,uis)(...)
121c  (onŁ(tli,(hi,thi),0),upd_hist(τ,tli)(upd_hist(τ,hi)(ahist))) end)
122. [119e] Or the automobile “disappears — off the radar”!

122 automobile_stop(ai)(aai,uis),(...)apos:atH(fli,hi,tli),ahist) ≡ stop

• Similar behaviour definitions can be given for automobiles on a link, for links and for hubs.

• Together they must reflect, amongst other things:
  – the time continuity of automobile flow,
  – that automobiles follow routes,
  – that automobiles, links and hubs together adhere to the intentional pull expressed earlier,
  – et cetera.

• A specification of these aspects must be proved to adhere to these properties.
6.4 Domain Behaviour Initialisation

• For every manifest part it must be described how its behaviour is initialised.

Example 62. The Road Transport System Initialisation: We “wrap up” the main example of this primer:

• We omit treatment of monitorable attributes.

123. Let us refer to the system initialisation as an action.
124. All links are initialised,
125. all hubs are initialised,
126. all automobiles are initialised,
127. etc.
value
123. \texttt{rts\_initialisation: \texttt{Unit} \rightarrow \texttt{Unit}}
124. \texttt{rts\_initialisation() \equiv}
125. \parallel \{ \text{link}(\text{uid}_L(l))(\text{meredo}_L(l))(\text{attr}_\text{LEN}(l))(\text{attr}_\text{\_Traffic}(l))(\text{attr}_\text{\_\Sigma}(l)) \mid l:L \cdot l \in ls \}
126. \parallel \{ \text{hub}(\text{uid}_H(l))(\text{meredo}_H(l))(\text{attr}_\text{\_\Omega}(l))(\text{attr}_\text{\_Traffic}(l))(\text{attr}_\text{\_\Sigma}(l)) \mid h:H \cdot h \in hs \}
127. \parallel \{ \text{automobile}(\text{uid}_A(a))(\text{meredo}_A(a))(\text{attr}_\text{\_RegNo}(a))(\text{attr}_\text{\_APos}(a)) \mid a:A \cdot a \in as \}
128. \parallel ...

• We have here omitted possible monitorable attributes.

• We refer to
  – \texttt{ls}: Item 36 on Slide 153,
  – \texttt{hs}: Item 37 on Slide 153, and
  – \texttt{as}: Item 38 on Slide 153
6.5 Discrete Dynamic Domains

• Up till now our analysis & description of a domain,
  – has, in a sense, been static:
  – in analysing a domain we considered its entities
  – to be of a definite number.

• In this section we shall consider the case
  where the number of entities change:
  – where new entities are created
  – and existing entities are destroyed,
  – that is:
    • where new parts, and hence behaviours, arise, and
    • existing parts, and hence behaviours, cease to exist.
6.5.1 Create and Destroy Entities

• In the domain we can expect that its behaviours create and destroy entities.

Example 63. Creation and Destruction of Entities:

• In the road transport domain
  – new hubs, links and automobiles may be inserted into the road net, and
  – existing links, hubs and automobiles may be removed from the road net.

• In a container terminal domain [11, 20]
  – new containers are introduced, old are discarded;
  – new container vessels are introduced, old are discarded;
  – new ship-to-shore cranes are introduced, old are discarded;
  – et cetera.
• In a *retailer* domain [22]
  – new customers are introduced, old are discarded;
  – new retailers are introduced, old are discarded;
  – new merchandise is introduced, old is discarded;
  – et cetera.

• In a *financial system* domain
  – new customers are introduced, old are discarded;
  – new banks are introduced, old are discarded;
  – new brokers are introduced, old are discarded;
  – et cetera
• **The issue here is:**

– When hubs and links are inserted or removed
  ∗ the mereologies of “neighbouring” road elements change,
  ∗ and so does the mereology of automobiles.

– When automobiles are inserted or removed
  ∗ The mereology of road elements
  ∗ have to be changed
  ∗ to take account of the insertions and removals,
  ∗ and so does the mereology of automobiles.

– And, some domain laws must be re-expressed:
  ∗ The domain part state, $\sigma$, must be updated$^3$,
  ∗ and so must the unique identifier state, $uid_\sigma^4$.

---

$^3$ Cf. Sect. 4.7.2 on Slide 155
$^4$ Cf. Sect. 5.2.4 on Slide 190
6.5.1.1 Create Entities

• It is taken for granted here that there are behaviours, one or more, which take the initiative to and carry out the creation of specific entities. Let us refer to such a behaviour as the “creator”.

• To create an entity implies the following three major steps
  – [A.–C.] the step wise creation of the part and initialisation of the transduced behaviour, and
  – [D.] the adjustment of all such part behaviours that might have their mereologies and attributes updated to accept such requests from creators.
A. To decide on the part sort – in order to create that part – that is
– to obtain a unique identifier – one hitherto not used;
– to obtain a mereology, one
  * according to the general mereology for parts of that sort,
  * and how the part specifically is to “fit” into its surroundings;
– to obtain an appropriate set of attributes:
  * again according to the attribute types for that part sort
  * and, more specifically, choosing initial attribute values.
– This part is then “joined” to \( \sigma^5 \) and
– its unique identifier “joined” to \( uid_\sigma^6 \).

\(^{(the \ global \ part \ state), \ Cf. \ Sect. \ 4.7.2 \ on \ Slide \ 155}\)
\(^{(the \ global \ unique \ identifier \ state), \ Cf. \ Sect. \ 5.2.4 \ on \ Slide \ 190}\)
B. Then to transcendentally deduce that part into a behaviour:
   – initialised (according to Sect. 6.3.1) with
     * the unique identifier,
     * the mereology, and
     * the attribute values
   – This behaviour is then invoked and “joined” to the set of current
     behaviours, cf. Sect. 6.4 on Slide 361 – i.e., just above!

C. Then, finally, to “adjust” the mereologies of topologically or
   conceptually related parts,
   – that is, for each of these parts to update:
   – their mereology and possibly some
   – state and state space
   arguments of their corresponding behaviours.
D. The update of the mereologies of already “running” behaviours requires the following:
– that, potentially all, behaviours offers to accept
– mereology update requests from the “creator” behaviour.

• The latter means, practically speaking,
  – that each part/behaviour
  – which may be subject to mereology changes
  – externally non-deterministically
  – expresses an offer to accept such a change.

Example 64. Road Net Administrator:
• We introduce the road net behaviour – based on the road net composite part, RN.
128. The road net has a programmable attribute: a road net 
(development & maintenance) graph.  

- The road net graph consists of a quadruple:
  - a map that for each hub identifier records “all” the
    information that the road net administrator deems necessary
    for the maintenance and development of road net hubs;
  - a map that for each link identifier records “all” the
    information that the road net administrator deems necessary
    for the maintenance and development of road net links;
  - and a map from the hub identifiers to the set of identifiers of
    the links it is connected to, and
  - the set of all automobile identifiers.

---

7 The presentation of the road net Behaviour, \( rn \), is simplified.
8 We presently abstract from what this information is.
9 See footnote 8.
129. This graph is commensurate with the actual topology of the road net.

type
128. \( G = (\text{HI} \to \text{H}_\text{Info}) \times (\text{LI} \to \text{L}_\text{Info}) \times (\text{HI} \to \text{L}-\text{set}) \times \text{AI}-\text{set} \)

value
128. \( \text{attr}_G: \text{RN} \rightarrow G \)

axiom
128. \( \forall (\text{hi}_\text{info}, \text{li}_\text{info}, \text{map}, \text{ais}): G \cdot \)
128. \( \text{dom} \ \text{map} = \text{dom} \ \text{hi}_\text{info} = \text{his} \land \lor \text{rng} \ \text{map} = \text{dom} \ \text{li}_\text{info} = \text{lis} \land \)
129. \( \forall \ \text{hi}: \text{HI} \cdot \ \text{hi} \in \text{dom} \ \text{hi}_\text{info} \implies \)
129. \( \text{let} \ \text{h}: \text{H} \cdot \ \text{h} \in \sigma \land \ \text{uid}_\text{H}(\text{h}) = \text{hi} \ \text{in} \)
129. \( \text{let} \ (\text{lis}',...') = \text{mereo}_\text{H}(\text{h}) \ \text{in} \ \text{lis}' = \text{map}(\text{hi}) \)
129. \( \text{ais} \subseteq \text{ais} \land \ldots \)
129. \( \text{end} \ \text{end} \)
• Please note the fundamental difference between
  – the road net (development & maintenance) graph and
  – the road net.
• The latter pretends to be “the real thing”.
• The former is “just” an abstraction thereof!
130. The road net mereology ("bypasses") the hub and link aggregates, and comprises a set of hub identifiers and a set of link identifiers – of the road net\(^{10}\).

\[
\text{type} \\
130. \text{H} \_ \text{Mer} = \text{Al-set} \times \text{LI-set} \\
130. \text{mereo}_\text{RN}: \text{RN} \rightarrow \text{RNMer} \\
\text{axiom} \\
130. \forall \text{rts:RTS} \cdot \text{let } (\_\_\_\_\_\_\_\_\text{lis}) = \text{mereo}_\text{H}(\text{obs}_\text{RN(rts)}) \text{ in lis } \subseteq \text{lis end} \\
\text{value}
\]

\(^{10}\)This is a repeat of the hub mereology given in Item 65 on Slide 210.
131. The road net [administrator] behaviour,
132. amongst other activities (…) 
133. internal non-deterministically decides upon

(a) either a hub insertion,
(b) or a link insertion,
(c) or a hub removal,
(d) or a link removal;

• These four sub-behaviours each resume being the road net behaviour.
\textbf{value}

131. \(\text{rn}: \text{RNI} \rightarrow \text{RNMer} \rightarrow \text{G} \rightarrow \text{in, out} \{ \text{ch}[\{i,j\}] \mid\{|i,j\}\subseteq uid_o}\}

131. \(\text{rn(rni)(rnmer)(g)} \equiv\)

132. ...

133a. \(\prod \text{insert_hub(g)(rni)(rnmer)}\)

133b. \(\prod \text{insert_link(g)(rni)(rnmer)}\)

133c. \(\prod \text{remove_hub(g)(rni)(rnmer)}\)

133d. \(\prod \text{remove_link(g)(rni)(rnmer)}\)
134. These road net sub-behaviours require information about
   (a) a hub to be inserted: its initial state, state space and [empty] traffic history, or
   (b) a link to be inserted: its length, initial state, state space and [empty] traffic history, or
   (c) a hub to be removed: its unique identifier, or
   (d) a link to be removed: its unique identifier.

   **type**
   134. \text{Info} == \text{nHInfo} | \text{nLInfo} | \text{oHInfo} | \text{oLInfo}
   134. \text{nHInfo} :: \text{H} \times \text{H} \times \text{H} \times \text{H}\_Traffic
   134. \text{nLInfo} :: \text{LEN} \times \text{L} \times \text{L} \times \text{L}\_Traffic
   134. \text{oHInfo} :: \text{HI}
   134. \text{oLInfo} :: \text{LI}
Example 65. **Road Net Development: Hub Insertion:**

- Road net development alternates between design, 
  - based on the *road net (development & maintenance) graph*, and 
- actual, “real life”, construction 
  - taking place in the real surroundings of the road net.
135. If a hub insertion then the road net behaviour, based on the hub and link information and the road net layout in the *road net (development & maintenance) graph* selects

(a) an initial mereology for the hub, \( h_{\text{mer}} \),
(b) an initial hub state, \( h_\sigma \), and
(c) an initial hub state space, \( h_\omega \), and
(d) an initial, i.e., empty hub traffic history;

136. updates its *road net (development & maintenance) graph* with information about the new hub,

137. and results in a suitable grouping of these.
value

135. design_new_hub: G → (nHInfo × G)
135. design_new_hub(g) ≡
135a. let h_mer: HMer = M_{ih}(g),
135b. hσ: HΣ = S_{ih}(g),
135c. hω: HΩ = O_{ih}(g),
135d. h_traffic = [],
136. g' = MSO_{ih}(g) in
137. ((h_mer, hσ, hω, h_traffic), g') end

- We leave open, in Items 135a–135c, as to what the initial hub mereology, state and state space should be initialised, i.e., the $M_{ih}, S_{ih}, O_{ih}$ and $MSO_{ih}$ functions.
138. To insert a new hub the road net administrator
   (a) first designs the new hub,
   (b) then selects a hub part
   (c) which satisfies the design,
       whereupon it updates the global states
   (d) of parts $\sigma$,
   (e) of unique identifiers, and
   (f) of hub identifiers –
       in parallel, and in parallel with
139. initiating a new hub behaviour
140. and resuming being the road net behaviour.
138. insert_hub: G×RNl×RNMer → Unit
138. insert_hub(g,rni,rnmer) ≡
138a. let ((h_mer,hσ,hω,h_traffic),g’) = design_new_hub(g) in
138b. let h:H · h∉σ ·
138c. mereo_H(h)=h_mer ∧ hσ=attr_H∑(h) ∧
138c. hω=attr_HΩ(h) ∧ h_traffic=attr_HTraffic(h) in
138d. σ := σ ∪ {h}
138e. || uid_σ := uid_σ ∪ {uid_H(h)}
138f. || his := his ∪ {uid_H(h)}
139. || hub(uid_H(h))(attr_H∑(h),attr_HΩ(h),attr_HΩ(h))
140. || rn(rni)(rnmer)(g’)
138. end end
Example 66. Road Net Development: Link Insertion:

141. If a link insertion then the road net behaviour based on the hub and link information and the road net layout in the *road net (development & maintenance) graph* selects

(a) the mereology for the link, \( h_{\text{mer}} \),
(b) the (static) length (attribute),
(c) an initial link state, \( l_\sigma \),
(d) an initial link state space \( l_\omega \), and
(e) and initial, i.e., empty, link traffic history;

142. updates its *road net (development & maintenance) graph* with information about the new link,

143. and results in a suitable grouping of these.

---

\(^{11}\) that is, the two existing hub identifiers between whose hubs the new link is to be inserted
value
141. design_new_link: G → (nLInfo×G)
141. design_new_link(g) ≡
141a. let l_mer:LMer = M_{il}(g),
141b. le:LEN = L_{il}(g),
141c. lσ:LΣ = S_{il}(g),
141d. lω:LΩ = O_{il}(g),
141e. l_hist:L_Hist = [ ]
142. g':G = MLSO_{il}(g) in
143. ((l_mer,le,lσ,lω,l_hist),g') end

• We leave open, in Items 141a–141d, as to what the initial link mereology, state and state space should be initialised.
144. To insert a new link the road net administrator
   (a) first designs the new link,
   (b) then selects a link part
   (c) which satisfies the design,
       whereupon it updates the global states
   (d) of parts, $\sigma$,
   (e) of unique part identifiers, and
   (f) of link identifiers –
       in parallel, and in parallel with
145. initiating a new link behaviour and
146. updating the mereologies and possibly the state and the state
    space attributes of the connected hubs.
value
144. insert_link: G → Unit
144. insert_link(rni,l) ≡
144a. let ((l_mer,le,lσ,lω,l_traffic_hist),g') = design_new_link(g) in
144c. let l:L · l∉σ · mereo_L(l)=l_mer ∧
144c. le=attr_LEN(l) ∧ lσ=attr_LΣ(l) ∧
144c. lω=attr_LΩ(l) ∧ l_traffic_hist=attr_HTraffic(l) in
144d. σ := σ ∪ {l}
144e. || uid_σ := uid_σ ∪ {uid_L(l)}
144f. || lis := list ∪ {}
145. || link(uid_L(l))(l_mer)(le,lω)(lσ,l_traffic)
146. || ch[ {rni,hi1} ] ! updH(M_{il}(g),Σ_{il}(g),Ω_{il}(g))
146. || ch[ {rni,hi2} ] !
144. end end ■

• We leave undefined the mereology and the state σ and state space ω update functions.
6.5.1.2 Destroy Entities

• The introduction to Sect. 6.5.1.1 on Slide 367 on the *creation of entities*
  – outlined a number of creation issues ([A, B, C, D]).
• For the *destruction of entities*
  – description matters are a bit simpler.
• It is, almost, simply a matter
  – of designating, by its unique identifier,
  – the entity: part and behaviour to be destroyed.
• Almost!
  – The mereology of the destroyed entity
  – must be such that the destruction
  – does not leave “dangling” references!
Example 67. **Road Net Development: Hub Removal:**

147. If a hub removal then the road net design_remove_hub behaviour, based on the *road net (development & maintenance) graph*, calculates the *unique hub identifier* of the “isolated” hub to be removed – that is, is not connected to any links,

148. updates the *road net (development & maintenance) graph*, and

149. results in a pair of these.

**value**

147. `design_remove_hub: G → (HI × G)`
147. `design_remove_hub(g) as (hi,g′)`
147. `let hi:HI • hi ∈ his ∧ let (_,lis) = mereo_H(retr_part(hi)) in lis={} end in`
148. `let g′ = M_{rh}(hi,g) in`
149. `(hi,g′) end end`
150. To remove a hub the road net administrator
   (a) first designs which old hub is to be removed
   (b) then removes the designated hub,
       whereupon it updates the global states
   (c) of parts $\sigma$,
   (d) of unique identifiers, and
   (e) of hub identifiers –
       in parallel, and in parallel with
151. stopping the old hub behaviour
152. and resuming being a road net behaviour.
value
150. remove_hub: G → RNI → RNMer → Unit
150. remove_hub(g)(rni)(rnmer) ≡
150a. let (hi,g′) = design_remove_hub(g) in
150b. let h:H ∙ uid_H(h)=hi ∧ ... in
150c. σ := σ \ {h}
150d. || uid_σ := uid_σ \ {hi}
150e. || his := his \ {hi}
151. || ch[{rni,hi}] ! mkStop()
152. || rn(rni)(rnmer)(g′)
150. end end ■
6.5.2 Adjustment of Creatable and Destructable Behaviours

• When an entity
  – is created or destroyed
  – its creation, respectively destruction
  – affects the mereologically related parts and their behaviours.
    ∗ their mereology
    ∗ and possibly their programmable state attributes
    ∗ need be adjusted.
  – And when entities are destroyed
    their behaviours are stopped!
  – These entities are “informed” so by the creator/destructor entity
    – as was shown in Examples 65–67.

• The next example will illustrate how such ‘affected’ entities
  handle such creator/destructor communication.
Example 68. **Hub Adjustments:**

- We have not yet illustrated hub (nor link) behaviours.
- Now we have to!

153. The mereology of a hub is a triple: the identification of the set of auromibles that may enter the hub, the identification of the set of links that connect to the hub, and the identification of the road net.

154. The hub behaviour external non-deterministically \(\lceil\rceil\rfloor\) alternates between

155. doing “own work”,

156. or accepting a stop “command” from the road net administrator, or

157. or accepting mereology & state update information, or

158. or other.
type
153. $H_{Mer} = \text{Al-set} \times \text{Ll-set} \times \text{RNI}$

value
153. mereo_H: $H \rightarrow H_{Mer}$
154. hub: $hi:Hl \rightarrow (\text{aulis}, \text{lis}, rni):H_{Mer} \rightarrow h\omega:H\Omega \rightarrow (h\sigma:H\Sigma \times ht:H\text{Traffic}) \rightarrow$
154. \hspace{1cm} \{ch[hi,ui]|ui:(\text{RNI}|\text{Al}) \cdot ui=\text{rni} \lor ui \in \text{aulis}\} \hspace{0.5cm} \text{Unit}
154. hub(hi)(hm:(\text{aulis}, \text{lis}, rni))(h\omega)(h\sigma,ht) \equiv 
155. ...
156. [] \text{let} \hspace{0.5cm} \text{mkStop()} = \text{ch}[hi, \text{rni}] \ ? \ \text{in} \ \text{stop} \ \text{end}
157. [] \text{let} \hspace{0.5cm} \text{mkUpdH}(hm', h\sigma', h\sigma') = \text{ch}[\{\text{rni}, hi\}] \ ? \ \text{in} 
157. \hspace{1cm} \text{hub(hi)(hm')(h\omega')(h\sigma',ht)} \ \text{end}
158. ...

• Observe from formula Item 156 that the hub behaviour ends,
• whereas “from” Item 157 it tail recurses! ■
6.5.3 Summary on Creatable & Destructable Entities

• We have sketched how we may model the dynamics of creating and destroying entities.
  – It is, but a sketch.
  – We should wish for a more methodological account.
  – So, that is what we are working on – amongst other issues – at the moment.
6.6 Domain Engineering: Description and Construction

- There are two meanings to the term ‘Domain Engineering’.
  – the construction of *descriptions* of domains, and
  – the construction of *domains*.

- Most sections of Chapters 4–6 are “devoted” to the former;
- the previous section, Sect. 6.5 to the latter.
6.7 Domain Laws

TO BE WRITTEN
6.8  A Domain Discovery Procedure, III

The predecessors of this section are Sects. 4.8.2 on Slide 159 and 5.6 on Slide 322.

6.8.1  Review of the Endurant Analysis and Description Process

• The discover... functions below were defined in Sects. 4.8.2 on Slide 159 and 5.6 on Slide 322.

value
endurant_analysis_and_description: Unit → Unit
endurant_analysis_and_description() ≡
    discover_sorts();  [Page 160]
    discover_internal_endurant_qualities()  [Page 324]

• We are now to define a perdurant_analysis_and_description procedure –
• to follow the above endurant_analysis_and_description procedure.
6.8.2 A Domain Discovery Process, III

- We define the `perdurant_analysis_and_description` procedure
  - in the reverse order of that of Sect. 5.6 on Slide 322,
  - first the full procedure,
  - then its sub-procedures.

```
A Domain Endurant Analysis and Description Process

value
perdurant_analysis_and_description: Unit \to Unit
perdurant_analysis_and_description() \equiv
discover_state();  \quad  \text{axiom} \ldots  \quad [\text{Note (a)}]
discover_channels();  \quad  \text{axiom} \ldots  \quad [\text{Note (b)}]
discover_behaviour_signatures();  \text{axiom} \ldots  \quad [\text{Note (c)}]
discover_behaviour_definitions();  \text{axiom} \ldots  \quad [\text{Note (d)}]
discover_initial_system()  \quad  \text{axiom} \ldots  \quad [\text{Note (e)}]
```

© Dines Bjørner, September 8, 2022
• Notes:
  – (a) **The States: \( \sigma \) and \( u_{i_{\sigma}} \)
    * We refer to Sect. 4.7.2 on Slide 155 and Sect. 5.2.4 on Slide 190.
    * The state calculation, as shown on Page 150, must be replicated, i.e., re-discovered, in any separate domain analysis & description.
    * The purpose of the state, i.e., \( \sigma \), is to formulate appropriate axiomatic constraints and domain laws.
  – (b) **The Channels:
    * We refer to Sects. 6.1.2 on Slide 338 and 6.2 on Slide 339.
    * Thus we indiscriminately declare a channel for each pair of distinct unique part identifiers whether the corresponding pair of part behaviours, if at all invoked, communicate or not.
(c) **Behaviour Signatures:**

* We refer to Sect. 6.3.1.2 on Slide 342.
* We find it more productive to first settle on the signatures of all behaviours – careful thinking has to go into that – before tackling the far more time-consuming work on defining the behaviours:

(d) **Behaviour Definitions:**

* We refer to Sect. 6.3.3 on Slide 352.

(e) **The Running System:**

* We refer to Sect. 6.4 on Slide 361.
### Summary

Perdurants: Analysis & Description

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Introduced</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Discovery Functions</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>discover_channels</td>
<td>page 339</td>
</tr>
<tr>
<td></td>
<td>discover_behaviour_signatures</td>
<td>page 351</td>
</tr>
<tr>
<td></td>
<td>discover_behaviour_definitions</td>
<td>page 354</td>
</tr>
<tr>
<td></td>
<td>discover_initial_system</td>
<td>page 361</td>
</tr>
<tr>
<td></td>
<td>perdurant_analysis_and_description</td>
<td>page 398</td>
</tr>
</tbody>
</table>
CHAPTER 7. Road Transport
7.1 The Road Transport Domain

- Our universe of discourse in this chapter is the road transport domain.

7.1.1 Naming

type RTS
7.1.2 Rough Sketch

• The road transport system that we have in mind consists of
  – a road net and
  – a set of vehicles
  – such that the road net serves to convey vehicles.

• We consider the road net to consist of
  – hubs, i.e., street intersections, or just street segment connection points, and
  – links, i.e., street segments between adjacent hubs.
• We consider vehicles to additionally include
  – departments of motor vehicles (DMVs),
  – bus companies, each with zero, one or more buses, and
  – vehicle associations, each with
    * zero, one or more members
    * who are owners of zero, one or more vehicles

---

1 This “rough” narrative fails to narrate what ...

© Dines Bjørner, September 8, 2022
7.2 External Qualities

A Road Transport System, I – Manifest External Qualities:

- Our intention is that the manifest external qualities of a road transport system are those of its
  - roads,
    * their hubs i.e., road (or street) intersections, and
    * their links, i.e., the roads (streets) between hubs, and
  - vehicles, i.e., automobiles – that ply the roads –
    * the buses, trucks, private cars, bicycles, etc.
7.2.1 A Road Transport System, II – Abstract External Qualities

- Examples of what could be considered abstract external qualities of a road transport domain are:
  - the aggregate of all hubs and all links,
  - the aggregate of all buses, say into bus companies,
  - the aggregate of all bus companies into public transport, and
  - the aggregate of all vehicles into a department of vehicles.

- Some of these aggregates may, at first be treated as abstract.

- Subsequently, in our further analysis & description we may decide to consider some of them as concretely manifested in, for example, actual
  - departments of roads.
7.2.2 Transport System Structure

• A transport system is modeled as structured into
  – a road net structure and
  – an automobile structure.

• The road net structure is then structured as a pair:
  – a structure of hubs and
  – a structure of links.

• These latter structures are then modeled as set of hubs, respectively links.
• We could have modeled the road net structure
  – as a *composite part*
  – with *unique identity, mereology and attributes*
  – which could then serve to model
  – a road net *authority*.

• And we could have modeled the automobile structure
  – as a *composite part*
  – with *unique identity, mereology and attributes*
  – which could then serve to model
  – a department of *vehicles*
7.2.3 Atomic Road Transport Parts

• From one point of view all of the following can be considered atomic parts:
  – hubs,
  – links\(^2\), and
  – automobiles.

\[\text{Hub} \equiv \text{street intersection}; \text{link} \equiv \text{street segments with no intervening hubs.}\]
7.2.4  Compound Road Transport Parts

7.2.4.1  The Composites

159. There is the *universe of discourse*, UoD.

   It is structured into

160. a *road net*, RN, and

161. a *fleet of vehicles*, FV.

   Both are structures. .................................

   **type**

   159  UoD  **axiom**  \( \forall uod:UoD \cdot \text{is\_structure}(uod) \).

   160  RN  **axiom**  \( \forall rn:RN \cdot \text{is\_structure}(rn) \).

   161  FV  **axiom**  \( \forall fv:FV \cdot \text{is\_structure}(fv) \).

   **value**

   160  obs\_RN: UoD \to RN

   161  obs\_FV: UoD \to FV
Figure 7.1: A Road Transport System Compounds and Structures
7.2.4.2 The Part Parts

162. The structure of hubs is a set, $sH$, of atomic hubs, $H$.
163. The structure of links is a set, $sL$, of atomic links, $L$.
164. The structure of buses is a set, $sBC$, of composite bus companies, $BC$.
165. The composite bus companies, $BC$, are sets of buses, $sB$.
166. The structure of private automobiles is a set, $sA$, of atomic automobiles, $A$. 
type
162  H, sH = H-set axiom ∀ h:H · is_atomic(h)
163  L, sL = L-set axiom ∀ l:L · is_atomic(l)
164  BC, BCs = BC-set axiom ∀ bc:BC · is_composite(bc)
165  B, Bs = B-set axiom ∀ b:B · is_atomic(b)
166  A, sA = A-set axiom ∀ a:A · is_atomic(a)

value
162  obs_sH: SH → sH
163  obs_sL: SL → sL
164  obs_sBC: SBC → BCs
165  obs Bs: BCs → Bs
166  obs_sA: SA → sA
7.2.5  The Transport System State

167. Let there be given a universe of discourse, $rts$. It is an example of a state.

From that state we can calculate other states.

168. The set of all hubs, $hs$.
169. The set of all links, $ls$.
170. The set of all hubs and links, $hls$.
171. The set of all bus companies, $bcs$.
172. The set of all buses, $bs$.
173. The set of all private automobiles, $as$.
174. The set of all parts, $ps$. 
value
167 \( rts:UoD \) \[34\]
168 \( hs:H\text{-}set \equiv H\text{-}set \equiv obs_{SH}(obs_{SH}(obs_{RN}(rts))) \)
169 \( ls:L\text{-}set \equiv L\text{-}set \equiv obs_{SL}(obs_{SL}(obs_{RN}(rts))) \)
170 \( hls:(H|L)\text{-}set \equiv hs\cup ls \)
171 \( bcs:BC\text{-}set \equiv obs_{BCs}(obs_{SBC}(obs_{FV}(obs_{RN}(rts)))) \)
172 \( bs:B\text{-}set \equiv \cup\{obs_{Bs}(bc)|bc:BC; bc \in bcs\} \)
173 \( as:A\text{-}set \equiv obs_{BCs}(obs_{SBC}(obs_{FV}(obs_{RN}(rts)))) \)
174 \( ps:(UoB|H|L|BC|B|A)\text{-}set \equiv rts\cup hls\cup bcs\cup bs\cup as \)
7.3 Internal Qualities

7.3.1 Unique Identifiers

175. We assign unique identifiers to all parts.

176. By a road identifier we shall mean a link or a hub identifier.

177. By a vehicle identifier we shall mean a bus or an automobile identifier.

178. Unique identifiers uniquely identify all parts.
   (a) All hubs have distinct [unique] identifiers.
   (b) All links have distinct identifiers.
   (c) All bus companies have distinct identifiers.
   (d) All buses of all bus companies have distinct identifiers.
   (e) All automobiles have distinct identifiers.
   (f) All parts have distinct identifiers.
type
175   H_UI, L_UI, BC_UI, B_UI, A_UI
176   R_UI = H_UI | L_UI
177   V_UI = B_UI | A_UI

value
178a  uid_H: H → H_UI
178b  uid_L: H → L_UI
178c  uid_BC: H → BC_UI
178d  uid_B: H → B_UI
178e  uid_A: H → A_UI
7.3.1.1 Extract Parts from Their Unique Identifiers

179. From the unique identifier of a part we can retrieve, $\varphi$, the part having that identifier.

type
179 $P = H \mid L \mid BC \mid B \mid A$

value
179 $\varphi : H_{UI} \rightarrow H \mid L_{UI} \rightarrow L \mid BC_{UI} \rightarrow BC \mid B_{UI} \rightarrow B \mid A_{UI} \rightarrow A$
179 $\varphi(ui) \equiv \textbf{let } p:(H\mid L\mid BC\mid B\mid A)p \in ps \land uid_P(p) = ui \textbf{ in } p \textbf{ end}$
7.3.1.2 All Unique Identifiers of a Domain

We can calculate:
180. the set, $h_{uis}$, of unique hub identifiers;
181. the set, $l_{uis}$, of unique link identifiers;
182. the map, $hl_{ui}m$, from unique hub identifiers to the set of unique link identifiers of the links connected to the zero, one or more identified hubs,
183. the map, $lh_{ui}m$, from unique link identifiers to the set of unique hub identifiers of the two hubs connected to the identified link;
184. the set, $r_{uis}$, of all unique hub and link, i.e., road identifiers;
185. the set, $bc_{uis}$, of unique bus company identifiers;
186. the set, $b_{ui}$s, of unique bus identifiers;
187. the set, $a_{ui}$s, of unique private automobile identifiers;
188. the set, $v_{ui}$s, of unique bus and automobile, i.e., vehicle identifiers;
189. the map, $bcb_{ui}m$, from unique bus company identifiers to the set of its unique bus identifiers; and
190. the (bijective) map, $bbc_{ui}bm$, from unique bus identifiers to their unique bus company identifiers.
value

180 \(h_{ui}s:H_{\text{UI-set}} \equiv \{\text{uid}_H(h)|h:H \land h \in hs\}\)

181 \(l_{ui}s:L_{\text{UI-set}} \equiv \{\text{uid}_L(l)|l:L \land l \in ls\}\)

184 \(r_{ui}s:R_{\text{UI-set}} \equiv h_{ui}s \cup l_{ui}s\)

182 \(hl_{ui}m:(H_{\text{UI}} \rightarrow L_{\text{UI-set}}) \equiv \)

182 \[h_{ui} \mapsto \text{luis}|h_{ui}:H_{\text{UI}}, \text{luis}:L_{\text{UI-set}} \land h_{ui}\in h_{ui}s \land (\_,\text{luis},\_) = \text{mergeo}_H(\eta(h_{ui}))\]

183 \(lh_{ui}m:(L_{+\text{UI}} \rightarrow H_{\text{UI-set}}) \equiv \)

183 \[l_{ui} \mapsto \text{huis}|h_{ui}:H_{\text{UI}}, \text{huis}:H_{\text{UI-set}} \land l_{ui}\in l_{ui}s \land (\_,\text{huis},\_) = \text{mergeo}_L(\eta(l_{ui}))\]

185 \(bc_{ui}s:BC_{\text{UI-set}} \equiv \{\text{uid}_BC(bc)|bc:BC \land bc \in bcs\}\)

186 \(b_{ui}s:B_{\text{UI-set}} \equiv \cup\{\text{uid}_B(b)|b:B \land b \in bs\}\)

187 \(a_{ui}s:A_{\text{UI-set}} \equiv \{\text{uid}_A(a)|a:A \land a \in as\}\)

188 \(v_{ui}s:V_{\text{UI-set}} \equiv b_{ui}s \cup a_{ui}s\)

189 \(bcb_{ui}m:(BC_{\text{UI}} \rightarrow B_{\text{UI-set}}) \equiv \)

189 \[bc_{ui} \mapsto \text{buis}|bc_{ui}:BC_{\text{UI}}, bc:BC \land bc \in bcs \land bc_{ui}=\text{uid}_BC(bc) \land (\_,\_,\text{buis})\]

190 \(bbc_{ui}bm:(B_{\text{UI}} \rightarrow BC_{\text{UI}}) \equiv \)

190 \[b_{ui} \mapsto bc_{ui}|b_{ui}:B_{\text{UI}}, bc_{ui}:BC_{\text{UI}} \land bc_{ui}=\text{dom}bcb_{ui}m \land b_{ui}\in bcb_{ui}m(bc_{ui})\]
7.3.1.3 Uniqueness of Road Net Identifiers

• We must express the following axioms:
191. All hub identifiers are distinct.
192. All link identifiers are distinct.
193. All bus company identifiers are distinct.
194. All bus identifiers are distinct.
195. All private automobile identifiers are distinct.
196. All part identifiers are distinct.

axiom
191 \( \text{card } hs = \text{card } h_{uis} \)
192 \( \text{card } ls = \text{card } l_{uis} \)
193 \( \text{card } bcs = \text{card } bc_{uis} \)
194 \( \text{card } bs = \text{card } b_{uis} \)
195 \( \text{card } as = \text{card } a_{uis} \)
196 \( \text{card } \{h_{uis} \cup l_{uis} \cup bc_{uis} \cup b_{uis} \cup a_{uis}\} = \text{card } h_{uis} + \text{card } l_{uis} + \text{card } bc_{uis} + \text{card } b_{uis} + \text{card } a_{uis} \)
7.3.2  Mereology

7.3.2.1  Mereology Types and Observers

197. The mereology of hubs is a pair: (i) the set of all bus and automobile identifiers, and (ii) the set of unique identifiers of the links that it is connected to and the set of all unique identifiers of all vehicles (buses and private automobiles).

198. 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.

199. The mereology of a bus company is a set the unique identifiers of the buses operated by that company.

200. The mereology of a bus is a pair: (i) the set of the one single unique identifier of the bus company it is operating for, and (ii) the unique identifiers of all links and hubs.

201. The mereology of an automobile is the set of the unique identifiers of all links and hubs.

<table>
<thead>
<tr>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>H_Mer</td>
<td>H_Ul-set × L_Ul-set</td>
</tr>
<tr>
<td>L_Mer</td>
<td>V_Ul-set × H_Ul-set</td>
</tr>
<tr>
<td>BC_Mer</td>
<td>B_Ul-set</td>
</tr>
<tr>
<td>B_Mer</td>
<td>BC_Ul-set × R_Ul-set</td>
</tr>
<tr>
<td>A_Mer</td>
<td>R_Ul-set</td>
</tr>
</tbody>
</table>

mereo_H: H → H_Mer
mereo_L: L → L_Mer
mereo_BC: BC → BC_Mer
mereo_B: B → B_Mer
mereo_A: A → A_Mer
7.3.2.2 Invariance of Mereologies

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

---

3 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.
4 The link identifiers designate the links, zero, one or more, that a hub is connected to is interested in the hub.
5 — that the bus might pass through
6 — that the automobile might pass through
7.3.2.2.1 **Invariance of Road Nets**

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

**axiom**

197. \( \forall (vuis,luis) : H_{\text{Mer}} \cdot luis \subseteq l_{uis} \land vuis = v_{uis} \)

198. \( \forall (vuis,huis) : L_{\text{Mer}} \cdot vuis = v_{uis} \land huis \subseteq h_{uis} \land \text{card}huis = 2 \)

199. \( \forall buis : H_{\text{Mer}} \cdot buis = b_{uis} \)

200. \( \forall (bc_{ui},ruis) : H_{\text{Mer}} \cdot bc_{ui} \subseteq bc_{uis} \land ruis = r_{uis} \)

201. \( \forall ruis : A_{\text{Mer}} \cdot ruis = r_{uis} \)
202. For all hubs, \( h \), and links, \( l \), in the same road net,
203. if the hub \( h \) connects to link \( l \) then link \( l \) connects to hub \( h \).

**axiom**

202 \( \forall h: H, l: L \cdot h \in hs \land l \in ls \Rightarrow \)
202 \quad \textbf{let} \ (\_,\text{luis}) = \text{mer eo}_H(h), (\_,\text{huis}) = \text{mer eo}_L(l)\)
203 \quad \textbf{in} \ \text{id}_L(l) \in \text{luis} \equiv \text{id}_H(h) \in \text{huis} \quad \textbf{end}
204. For all links, \( l \), and hubs, \( h_a, h_b \), in the same road net, 
205. if the \( l \) connects to hubs \( h_a \) and \( h_b \), then \( h_a \) and \( h_b \) both connects to 
link \( l \).

axiom

\[ \forall h_a, h_b : H, l : L \cdot \{h_a, h_b\} \subseteq hs \land l \in ls \Rightarrow \]

\[ \text{let } (_{luis}) = \text{mereo}_H(h), (_{huis}) = \text{mereo}_L(l) \]

\[ \text{in } uid_L(l) \in luis \equiv uid_H(h) \in huis \text{ end} \]
7.3.2.2.2 Possible Consequences of a Road Net Mereology

206. are there [isolated] units from which one can not “reach” other units?

207. does the net consist of two or more “disjoint” nets?

208. et cetera.

• We leave it to the reader to narrate and formalise the above properly.
7.3.2.2.3 Fixed and Varying Mereology

• Let us consider a road net.
  – 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 is a fixed mereology.
– 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 is a *varying mereology*. 
7.3.3 Attributes

7.3.3.1 Hub Attributes

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

209. 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 from link \(l_f\) to 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.
210. 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.

211. 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
209 \( H\Sigma = (L_{UI} \times L_{UI})\)-set

axiom
209 \( \forall h:H \cdot obs_{H\Sigma}(h) \in obs_{H\Omega}(h) \)

type
210 \( H\Omega = H\Sigma\)-set
211 \( H_{Traffic} \)
211 \( H_{Traffic} = (A_{UI} \mid B_{UI}) \rightarrow (TIME \times VPos)^* \)

axiom
211 \( \forall ht:H_{Traffic}, ui:(A_{UI} \mid B_{UI}) \cdot \)
211 \( ui \in \text{dom} \, ht \Rightarrow \text{time\_ordered}(ht(ui)) \)

value
209 attr_{H\Sigma}: H \rightarrow H\Sigma
210 attr_{H\Omega}: H \rightarrow H\Omega
211 attr_{H_{Traffic}}: H \rightarrow H_{Traffic}

value
211 time\_ordered: (TIME \times VPos)^* \rightarrow Bool
211 time\_ordered(tvpl) \equiv ...
7.3.3.2 Invariance of Traffic States

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

**axiom**

212 $\forall h : H \cdot h \in hs \Rightarrow$

212 let $h\sigma = \text{attr}_H\Sigma(h)$ in

212 $\forall (l_{ui}i, l_{ui}i') : (L_{UI} \times L_{UI}) \cdot (l_{ui}i, l_{ui}i') \in h\sigma \Rightarrow \{l_{ui}i, l_{ui}i'\} \subseteq l_{ui}s \text{ end}$
7.3.3.3 Link Attributes

We show just a few attributes.

213. 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 from hub \(h_f\) to hub \(h_t\). Link states can have either 0, 1 or 2 elements.

214. 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 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".
215. 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.

216. The hub identifiers of link states must be in the set, $h_{ui}$, of the road net’s hub identifiers.
type
213 \( L_\Sigma = H_{UI}\)-set

axiom
213 \( \forall \sigma : L_\Sigma \cdot \text{card } \sigma = 2 \)
213 \( \forall l : L \cdot \text{obs}_L(l) \in \text{obs}_L(\Omega)(l) \)

type
214 \( L \Omega = L_\Sigma\)-set
215 \( L_{\text{Traffic}} \)
215 \( L_{\text{Traffic}} = (A_{UI} \mid B_{UI}) \rightarrow (\mathbb{T} \times (H_{UI} \times \text{Frac} \times H_{UI}))^* \)
215 \( \text{Frac} = \text{Real}, \text{axiom frac:Fract } \cdot 0 < \text{frac} < 1 \)

value
213 \( \text{attr}_\Sigma : L \rightarrow L_\Sigma \)
214 \( \text{attr}_\Omega : L \rightarrow L \Omega \)
215 \( \text{attr}_\text{Traffic}: : \rightarrow L_{\text{Traffic}} \)

axiom
215 \( \forall \text{lt:L}_{\text{Traffic}}, ui : (A_{UI} \mid B_{UI}) \cdot \text{ui} \in \text{dom} \ ht \Rightarrow \text{time}_\text{ordered}(ht(ui)) \)
216 \( \forall l : L \cdot l \in ls \Rightarrow \)
216 \( \text{let } \sigma = \text{attr}_\Sigma(l) \text{ in } \forall (h_{ui,i}, h_{ui,i}') : (H_{UI} \times K_{UI}) \cdot \)
216 \( (h_{ui,i}, h_{ui,i}') \in l_\sigma \Rightarrow \{h_{ui,i}, h_{ui,i}'\} \subseteq h_{ui}s \text{ end } \)
7.3.3.4  **Bus Company Attributes**

- Bus companies operate a number of lines that service passenger transport along routes of the road net. Each line being serviced by a number of buses.

217. Bus companies create, maintain, revise and distribute [to the public (not modeled here), and to buses] bus time tables, not further defined.

**type**

217  BusTimTbl

**value**

217  attr_BusTimTbl: BC → BusTimTbl

- There are two notions of time at play here:
  - the indefinite “real” or “actual” time; and
  - the definite calendar, hour, minute and second time designation occurring in some textual form in, e.g., time tables.
7.3.3.5  Bus Attributes

We show just a few attributes.

218. Buses run routes, according to their line number, ln:LN, in the
219. bus time table, btt:BusTimTbl obtained from their bus company,
    and and keep, as inert attributes, their segment of that time table.
220. Buses occupy positions on the road net:
    (a) either at a hub identified by some h_ui,
    (b) or on a link, some fraction, f: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.
221. Et cetera.
type
218  LN
219  BusTimTbl
220  BPos  == atHub | onLink
220a  atHub  :: h Ui:H Ul
220b  onLink  :: fh Ui:H Ul×l Ui:L Ul×frac:Fract×th Ui:H Ul
220b  Fract  = Real, axiom  frac:Fract · 0<frac<1
221  ...

value
219  attr_BusTimTbl: B → BusTimTbl
220  attr_BPos: B → BPos
7.3.3.6 Private Automobile Attributes

• We illustrate but a few attributes:

222. Automobiles have static number plate registration numbers.

223. Automobiles have dynamic positions on the road net:

[220a] either at a hub identified by some \( h_{ui} \),

[220b] 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, \( th_{ui} \).
type

222 RegNo
def APos == atHub | onLink

220a atHub :: h_ui:H_Ui
220b onLink :: fh_ui:H_Ui × l_ui:L_Ui × frac:Fract × th_ui:H_Ui
220b Fract = Real, axiom frac:Fract \cdot 0 < frac < 1

value

222 attr_RegNo: A → RegNo
223 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.
• Observe that bus companies each have their own distinct *bus time table*, and that these are modeled as *programmable*, Item 217 on Slide 439, page 439.

• Observe then that buses each have their own distinct *bus time table*, and that these are model-led as *inert*, Item 219 on Slide 440, page 440.
• In Items Sli. 257 and Sli. 261, 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.⁷

---

⁷In 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.
7.3.3.7 Intentionality

224. Seen from the point of view of an automobile there is its own traffic history, $A_{\text{Hist}}$, which is a (time ordered) sequence of timed automobile’s positions;

225. seen from the point of view of a hub there is its own traffic history, $H_{\text{Traffic}}$ Item Sli. 257, which is a (time ordered) sequence of timed maps from automobile identities into automobile positions; and

226. seen from the point of view of a link there is its own traffic history, $L_{\text{Traffic}}$ Item Sli. 261, which is a (time ordered) sequence of timed maps from automobile identities into automobile positions.

• The intentional “pull” of these manifestations is this:

227. The union, i.e. proper merge of all automobile traffic histories, $\text{AllATH}$, must now be identical to the same proper merge of all hub, $\text{AllHTH}$, and all link traffic histories, $\text{AllLTH}$. 
type
224  \text{A}_{\text{Hi}} = (T \times \text{APos})^* \\
211  \text{H}_{\text{Trf}} = \text{A}_{\text{UI}} \rightarrow (\text{TIME} \times \text{APos})^* \\
215  \text{L}_{\text{Trf}} = \text{A}_{\text{UI}} \rightarrow (\text{TIME} \times \text{APos})^* \\
227  \text{AllATH} = \text{TIME} \rightarrow (\text{AUI} \rightarrow \text{APos}) \\
227  \text{AllHTH} = \text{TIME} \rightarrow (\text{AUI} \rightarrow \text{APos}) \\
227  \text{AllLTH} = \text{TIME} \rightarrow (\text{AUI} \rightarrow \text{APos}) \\

axiom
227  \text{let } \text{allA} = \text{mrg}_{\text{AllATH}}(\{(a, \text{attr}_{\text{A}_{\text{Hi}}}(a)) | a : A \cdot a \in as\}), \\
227  \text{allH} = \text{mrg}_{\text{AllHTH}}(\{\text{attr}_{\text{H}_{\text{Trf}}}(h) | h : H \cdot h \in hs\}), \\
227  \text{allL} = \text{mrg}_{\text{AllLTH}}(\{\text{attr}_{\text{L}_{\text{Trf}}}(l) | l : L \cdot h \in ls\}) \text{ in } \\
227  \text{allA} = \text{mrg}_{\text{HLT}}(\text{allH, allL}) \text{ end}
• We leave the definition of the merge functions to the listener!
  – We endow
    * each automobile with its history of timed positions and
    * each hub and link with their histories of timed automobile positions.
  – These histories are facts!
  – They are not something that is laboriously recorded, where such recordings may be imprecise or cumbersome.
  – The facts are there, so we can (but may not necessarily) talk about these histories as facts.

---

-or thought technologically in-feasible – at least some decades ago!
– It is in that sense that the purpose (`transport`) for which man let automobiles, hubs and link be made with their `transport` intent are subject to an intentional “pull”.

• *It can be no other way: if automobiles “record” their history, then hubs and links must together “record” identically the same history!.*
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*
7.4 Perdurants

- In this section we transcendently “morph” parts into behaviours.
- We analyse that notion and its constituent notions of
  - actors,
  - channels and communication,
  - actions and
  - events.
- The main transcendental deduction of this chapter
  - is that of associating
  - with each part
  - a behaviour.
- This section shows the details of that association.
• Perdurants are understood in terms of
  – a notion of \textit{state} and
  – a notion of \textit{time}. 
State Values versus State Variables:

- Item 174 on Slide 415 expresses the value of all parts of a road transport system:

\[ ps: (UoB|H|L|BC|B|A) - set \equiv rts \cup hls \cup bcs \cup bs \cup as. \]

228. We now introduce the set of variables, one for each part value of the domain being modeled.

228. \{ \textbf{variable} \: vp:(UoB|H|L|BC|B|A) \mid vp:(UoB|H|L|BC|B|A) \cdot vp \in ps \}
Buses and Bus Companies

• A bus company is like a “root” for its fleet of “sibling” buses.
• But a bus company may cease to exist without the buses therefore necessarily also ceasing to exist.
• They may continue to operate, probably illegally, without, possibly.
  a valid bus driving certificate.
• Or they may be passed on to either private owners or to other bus companies.
• We use this example as a reason for not endowing a “block structure” concept on behaviours.
7.4.1 Channels and Communication

7.4.1.1 Channel Message Types

- We ascribe types to the messages offered on channels.

229. Hubs and links communicate, both ways, with one another, over channels, h1_ch, whose indexes are determined by their mereologies.

230. Hubs send one kind of messages, links another.

231. Bus companies offer timed bus time tables to buses, one way.

232. Buses and automobiles offer their current, timed positions to the road element, hub or link they are on, one way.

\[
\begin{align*}
\text{type} & \quad 230 \text{ H_L_Msg, L_H_Msg} \\
& \quad 229 \text{ H_L_Msg} = \text{H_L_Msg} \mid \text{L_F_Msg} \\
& \quad 231 \text{ B_C_B_Msg} = T \times \text{BusTimTbl} \\
& \quad 232 \text{ V_R_Msg} = T \times (BPos|APos)
\end{align*}
\]
7.4.1.2 Channel Declarations

233. This justifies the channel declaration which is calculated to be:

\[
\text{channel}
\]

\[
233 \{ \text{hl}_\text{ch}[h_{ui},l_{ui}]:H_{L}\_\text{Msg} \mid h_{ui}:H_{UL},l_{ui}:L_{UL} | i \in h_{ui}s \land j \in l_{ui}m(h_{ui}) \}
\]

\[
233 \cup
\]

\[
233 \{ \text{hl}_\text{ch}[h_{ui},l_{ui}]:L_{H}\_\text{Msg} \mid h_{ui}:H_{UL},l_{ui}:L_{UL} | l_{ui} \in l_{ui}s \land i \in l_{ui}m(l_{ui}) \}
\]
• We shall argue for bus company-to-bus channels based on the mereologies of those parts.
  – Bus companies need communicate to all its buses, but not the buses of other bus companies.
  – Buses of a bus company need communicate to their bus company, but not to other bus companies.

234. This justifies the channel declaration which is calculated to be:

channel 234 \{ bc\_b\_ch[bc\_ui,b\_ui] \mid bc\_ui:BC\_UI, b\_ui:B\_UI \cdot bc\_ui \in bc\_ui:s \land b\_ui \in b\_ui:s \}: BC\_B\_Msg
• We shall argue for vehicle to road element channels based on the mereologies of those parts.
  – Buses and automobiles need communicate to
    * all hubs and
    * all links.

235. This justifies the channel declaration which is calculated to be:

\[
\text{channel} \quad 235 \quad \{ v_{r\_ch}[v_{ui},r_{ui}] \mid v_{ui} : V\_UL, r_{ui} : R\_UL \cdot v_{ui} \in v_{ui}s \land r_{ui} \in r_{ui}s \} : V\_R\_Msg
\]
7.4.2 Behaviours

7.4.2.1 Road Transport Behaviour Signatures

• We first decide on names of behaviours.
  – In the translation schemas
  – we gave schematic names to behaviours
  – of the form $\mathcal{M}_p$.

• We now assign mnemonic names:
  – from part names to names of transcendentally interpreted behaviours
  – and then we assign signatures to these behaviours.
7.4.2.1.1 Hub Behaviour Signature

236. hub_{h_{ui}}:

(a) there is the usual “triplet” of arguments: unique identifier, mereology and static attributes;
(b) then there are the programmable attributes;
(c) and finally there are the input/output channel references: first those allowing communication between hub and link behaviours,
(d) and then those allowing communication between hub and vehicle (bus and automobile) behaviours.
value
236  hub$^h_{ui}$:
236a  $h_{ui} : H_{UI} \times (vuis, luis, \_): H_{Mer} \times H_{\Omega}$
236b  $\rightarrow (H_{\Sigma} \times H_{\text{Traffic}})$
236c  $\rightarrow \textbf{in, out} \{ h_{l_{ch}}[h_{ui}, l_{ui}] | l_{ui} : L_{UI} | l_{ui} \in luis \}$
236d  $\{ \text{ba}_{r_{ch}}[h_{ui}, v_{ui}] | v_{ui} : V_{UI} | v_{ui} \in vuis \}$ \textbf{Unit}
236a  \textbf{pre}: $\text{vuis} = v_{ui} s \land \text{luis} = l_{ui} s$
7.4.2.1.2  Link Behaviour Signature

237. \( \text{link}_{lui} \):

(a) there is the usual “triplet” of arguments: unique identifier, mereology and static attributes;
(b) then there are the programmable attributes;
(c) and finally there are the input/output channel references: first those allowing communication between hub and link behaviours,
(d) and then those allowing communication between link and vehicle (bus and automobile) behaviours.
value
237  \text{link}_{l_{ui}} : \\
237a  l_{ui} \in \text{L}_{UI} \times (\text{vuis}, \text{huis}, \_): \text{L}_{Mer} \times \text{L}_{\Omega} \\
237b  \rightarrow (L_{\Sigma} \times \text{L}_{\text{Traffic}}) \\
237c  \rightarrow \text{in, out} \\{ h_{\_\text{l}\_\text{ch}}[h_{ui}, l_{ui}] \mid h_{ui} \in \text{H}_{UI} : h_{ui} \in \text{huis} \} \\
237d  \\{ ba\_r\_ch[l_{ui}, v_{ui}] \mid v_{ui} : (B_{UI} \mid A_{UI}) • v_{ui} \in \text{vuis} \} \textbf{Unit} \\
237a  \textbf{pre: suis} = v_{ui}s \land \text{huis} = h_{ui}s
7.4.2.1.3 Bus Company Behaviour Signature

238. \textit{bus\_company}_{bc_{ui}}:

(a) there is here just a “doublet” of arguments: unique identifier and mereology;

(b) then there is the one programmable attribute;

(c) and finally there are the input/output channel references allowing communication between the bus company and buses.
value

238 \text{bus\_company}_{bc_{ui}}:
238a \text{bc\_ui:BC\_UI}\times(\_\_\_,\_\_\_,\text{buis}):\text{BC\_Mer}
238b \rightarrow \text{BusTimTbl}
238c \text{in},\text{out} \{\text{bc\_b\_ch[bc\_ui,b\_ui]}|b\_ui:B\_Uib\_ui\in\text{buis}\} \textbf{Unit}
238a \textbf{pre}: \text{buis} = b_{ui}s \land \text{huis} = h_{ui}s
7.4.2.1.4  Bus Behaviour Signature

239. bus\(_{\text{bus}}\):

(a) there is here just a “doublet” of arguments: unique identifier and mereology;

(b) then there are the programmable attributes;

(c) and finally there are the input/output channel references: first the input/output allowing communication between the bus company and buses,

(d) and the input/output allowing communication between the bus and the hub and link behaviours.
value
239  \text{bus}_b_{ui}:
239a  \text{b}_{ui}: B_{UI} \times (bc_{ui}, _{ruis}): B_{Mer}
239b  \rightarrow (LN \times BTT \times BPOS)
239c  \rightarrow \textbf{out} bc_{b_ch}[bc_{ui},b_{ui}],
239d  \{ \text{bar}_{ch}[r_{ui},b_{ui}][r_{ui}:(H_{UI}| L_{UI})_{ui} \in v_{ui}s] \textbf{Unit} \}
239a  \textbf{pre}: \text{ruis} = r_{ui}s \land bc_{ui} \in bc_{ui}s
7.4.2.1.5 **Automobile Behaviour Signature**

240. automobile$_{a_{ui}}$:

(a) there is the usual “triplet” of arguments: unique identifier, mereology and static attributes;

(b) then there is the one programmable attribute;

(c) and finally there are the input/output channel references allowing communication between the automobile and the hub and link behaviours.
value
240  automobile_{a_{ui}}:
240a  a_{ui}:A_{UI}\times(_,_,ruis):A\_Mer\times r\_n:RegNo
240b  \rightarrow apos:APos
240c  in,out \{ba_{\_ch}[a_{ui},r_{ui}]|r_{ui}:(H_{UI}|L_{UI})\_r_{ui} \in ruis\} Unit
240a  pre: ruis = r_{ui}s \land a_{ui} \in a_{ui}s
7.4.2.2  Behaviour Definitions

• We only illustrate automobile, hub and link behaviours.

7.4.2.2.1  Automobile Behaviour at a Hub

• We define the behaviours in a different order than the treatment of their signatures.

• We “split” definition of the automobile behaviour
  – into the behaviour of automobiles when positioned at a hub, and
  – into the behaviour automobiles when positioned at on a link.

• In both cases the behaviours include the “idling” of the automobile, i.e., its “not moving”, standing still.
241. We abstract automobile behaviour at a Hub (hui).
242. The vehicle remains at that hub, “idling”,
243. informing the hub behaviour,
244. or, internally non-deterministically,
   (a) moves onto a link, tl_i, whose “next” hub, identified by th_ui, is obtained from the mereology of the link identified by tl_ui;
   (b) informs the hub it is leaving and the link it is entering of its initial link position,
   (c) whereupon the vehicle resumes the vehicle behaviour positioned at the very beginning (0) of that link,
245. or, again internally non-deterministically,
246. the vehicle “disappears — off the radar”!
automobile\textsubscript{a\_ui}(a\_ui, (\{\}, (ruis, vuis), \{\}), rn)\)

\(\text{apos}:\text{atH}(fl\_ui, h\_ui, tl\_ui) \equiv \)

\((\text{ba\_r\_ch}[a\_ui, h\_ui])! (\text{record\_TIME}(), \text{atH}(fl\_ui, h\_ui, tl\_ui));\)

automobile\textsubscript{a\_ui}(a\_ui, (\{\}, (ruis, vuis), \{\}), rn)(apos)\)

\(\parallel\)

\((\text{let} (\{fh\_ui, th\_ui\}, ruis\') = \text{mero\_L}(\wp(tl\_ui)) \text{ in}\)

\assert: fh\_ui = h\_ui \land ruis = ruis\')\)

\((\text{let} onl = (tl\_ui, h\_ui, 0, th\_ui) \text{ in}\)

\((\text{ba\_r\_ch}[a\_ui, h\_ui])! (\text{record\_TIME}(), \text{onL}(onl)) \mid\)

\((\text{ba\_r\_ch}[a\_ui, tl\_ui])! (\text{record\_TIME}(), \text{onL}(onl))) ;\)

automobile\textsubscript{a\_ui}(a\_ui, (\{\}, (ruis, vuis), \{\}), rn)\)

\((\text{onL}(onl)) \text{ end end})\)

\(\parallel\)

\(\text{stop}\)
Automobile Behaviour On a Link

We abstract automobile behaviour on a Link.

(a) Internally non-deterministically, either
   i. the automobile remains, “idling”, i.e., not moving, on the link,
   ii. however, first informing the link of its position,

(b) or
   i. if if the automobile’s position on the link has not yet reached the hub, then
      A. then the automobile moves an arbitrary small, positive Real-valued increment along the link
      B. informing the hub of this,
      C. while resuming being an automobile ate the new position,
ii. else,
   A. while obtaining a “next link” from the mereology of the hub (where that next link could very well be the same as the link the vehicle is about to leave),
   B. the vehicle informs both the link and the imminent hub that it is now at that hub, identified by th_ui,
   C. whereupon the vehicle resumes the vehicle behaviour positioned at that hub;
   (c) or
   (d) the vehicle “disappears — off the radar”!
automobile_{a_{ui}}(a_{ui},([],ruis,[]),rno) 
\text{vp:onL}((fh_{ui},l_{ui},f,th_{ui})) \equiv 
\begin{align*}
\text{ba_r_ch}[thui,a_{ui}]!atH(lui,thui,nxt_{lui}) ; \\
\text{automobile}_{a_{ui}}(a_{ui},([],ruis,[]),rno)(vp)
\end{align*} 
\begin{align*}
\text{if not.yet.at_hub}(f) \\
\text{then} \\
\begin{align*}
\text{let incr = increment}(f) \text{ in} \\
\text{let onl = (tl_{ui},h_{ui},incr,th_{ui}) in} \\
\text{ba_r_ch}[l_{ui},a_{ui}]!onL(onl) ; \\
\text{automobile}_{a_{ui}}(a_{ui},([],ruis,[]),rno) \\
\text{(onL(onl))}
\end{align*} \\
\text{end end) }
\begin{align*}
\text{else} \\
\begin{align*}
\text{let nxt_{lui}:L_{UL}} & \cdot \text{nxt_{lui}} \in \text{mereo}_H(\wp(th_{ui})) \text{ in} \\
\text{ba_r_ch}[thui,a_{ui}]!atH(l_{ui},th_{ui},nxt_{lui}) ; \\
\text{automobile}_{a_{ui}}(a_{ui},([],ruis,[]),rno) \\
\text{(atH(l_{ui},th_{ui},nxt_{lui})) end)
\end{align*}
\end{align*} \\
\text{end) }
\begin{align*}
\text{else in}
\begin{align*}
\text{increment: Fract \rightarrow Fract}
\end{align*}
\end{align*}
7.4.2.2.3 Hub Behaviour

248. The hub behaviour

(a) non-deterministically, externally offers
(b) to accept timed vehicle positions —
(c) which will be at the hub, from some vehicle, \( v_{ui} \).
(d) The timed vehicle hub position is appended to the front of that vehicle’s entry in the hub’s traffic table;
(e) whereupon the hub proceeds as a hub behaviour with the updated hub traffic table.
(f) The hub behaviour offers to accept from any vehicle.
(g) A post condition expresses what is really a proof obligation: that the hub traffic, \( ht' \) satisfies the axiom of the endurant hub traffic attribute Item Sli. 257.

\[
\begin{align*}
\text{value} \\
\text{hub}_{hui}(h_{ui},((luis,vuis)),h\omega)(h\sigma,ht) \equiv \\
\text{let } m = \text{barch}[h_{ui},v_{ui}] \text{ ? in} \\
\text{assert: } m = \text{atHub}(\_,h_{ui},\_)) \\
\text{let } ht' = ht \uplus \{ h_{ui} \mapsto \langle m \rangle \text{ht}(h_{ui}) \} \text{ in} \\
\text{hub}_{hui}(h_{ui},((luis,vuis)),(h\omega))(h\sigma,ht') \\
\text{let } v_{ui}:V_{UI} \in \text{vuis} \text{ end end } \\
\text{post: } \forall v_{ui}:V_{UI} v_{ui} \in \text{dom} \text{ ht'\Rightarrowtime_ordered(ht'(v_{ui}))}
\end{align*}
\]
7.4.2.2.4 Link Behaviour

249. The link behaviour non-deterministically, externally offers
250. to accept timed vehicle positions —
251. which will be on the link, from some vehicle, \( v_{ui} \).
252. The timed vehicle link position is appended to the front of that vehicle’s entry in the link’s
traffic table;
253. whereupon the link proceeds as a link behaviour with the updated link traffic table.
254. The link behaviour offers to accept from any vehicle.
255. A post condition expresses what is really a proof obligation: that the link traffic, \( lt' \) satisfies the
axiom of the endurant link traffic attribute Item Sli. 261.

\[
\text{link}_{lvui}(l_{ui},(\_,(huis,vuis),\_),l_{\omega})(l_{\sigma},lt) \equiv \\
\begin{cases}
\{ \text{let } m = \text{ba_r_ch}[l_{ui},v_{ui}] ? \text{ in} \\
\quad \text{assert: } m = (\_,\text{onLink}(\_,l_{ui},\_,\_)) \\
\quad \text{let } lt' = lt \uplus \{ l_{ui} \mapsto \langle m \rangle \hat{\text{lt}}(l_{ui}) \} \text{ in} \\
\quad \text{link}_{lvui}(l_{ui},(huis,vuis),h_{\omega})(h_{\sigma},lt') \\
\quad \text{end end} \}
\end{cases}
\]

255 post: \( \forall v_{ui}:V_{UI} \cdot v_{ui} \in \text{dom } lt' \Rightarrow \text{time_ordered}(lt'(v_{ui})) \)
7.5  System Initialisation

7.5.1  Initial States

value

\[ hs: \text{H-set} \equiv \text{obs}_sH(\text{obs}_S\text{H}(\text{obs}_RN(rts))) \]
\[ ls: \text{L-set} \equiv \text{obs}_sL(\text{obs}_SL(\text{obs}_RN(rts))) \]
\[ bcs: \text{BC-set} \equiv \text{obs}_sBC(\text{obs}_S\text{BC}(\text{obs}_FV(\text{obs}_RN(rts)))) \]
\[ bs: \text{B-set} \equiv \bigcup\{\text{obs}_sB(bc) | bc: \text{BC}, bc \in bcs\} \]
\[ as: \text{A-set} \equiv \text{obs}_sBCs(\text{obs}_S\text{BC}(\text{obs}_FV(\text{obs}_RN(rts)))) \]
7.5.2 **Initialisation**

- We are reaching the end of this domain modeling example.
  - Behind us there are narratives and formalisations.
  - Based on these we now express
    * the signature and
    * the body of the definition
  - of a “system build and execute” function.

256. The system to be initialised is

(a) the parallel compositions (||) of
(b) the distributed parallel composition (||[...][...]) of all hub behaviours,
(c) the distributed parallel composition (||[...][...]) of all link behaviours,
(d) the distributed parallel composition (||[...][...]) of all bus company behaviours,
(e) the distributed parallel composition (||[...][...]) of all bus behaviours, and
(f) the distributed parallel composition (||[...][...]) of all automobile behaviours.
value
256 initial_system: Unit → Unit
256 initial_system() ≡
256b || { hub\textsubscript{hub}(h\textsubscript{ui},me,h\omega)(h\textsubscript{trf},h\sigma)
256b | h:H h ∈ hs, h\textsubscript{ui}:H\textsubscript{UI} h\textsubscript{ui}=uid\textsubscript{H}(h), me:HMet\textsubscript{L} me=mere\textsubscript{L}(h),
256b h\textsubscript{trf}:H\textsubscript{Traffic} h\textsubscript{trf}=attr\textsubscript{H\textsubscript{Traffic}}(h),
256b h\omega:H\textsubscript{Ω} h\omega=attr\textsubscript{H\textsubscript{Ω}}(h), h\sigma:H\textsubscript{Σ} h\sigma=attr\textsubscript{H\textsubscript{Σ}}(h) ∧ h\sigma ∈ h\omega }
256a ||
256c || { link\textsubscript{link}(l\textsubscript{ui},me,l\omega)(l\textsubscript{trf},l\sigma)
256c l:L l ∈ ls, l\textsubscript{ui}:L\textsubscript{UI} l\textsubscript{ui}=uid\textsubscript{L}(l), me:LMet\textsubscript{L} me=mere\textsubscript{L}(l),
256c l\textsubscript{trf}:L\textsubscript{Traffic} l\textsubscript{trf}=attr\textsubscript{L\textsubscript{Traffic}}(l),
256c l\omega:L\textsubscript{Ω} l\omega=attr\textsubscript{L\textsubscript{Ω}}(l), l\sigma:L\textsubscript{Σ} l\sigma=attr\textsubscript{L\textsubscript{Σ}}(l) ∧ l\sigma ∈ l\omega }
256a ||
256d || { bus\textsubscript{company}\textsubscript{bcui}(bc\textsubscript{ui},me)(btt)
256d bc:BC bc ∈ bcs, bc\textsubscript{ui}:BC\textsubscript{UI} bc\textsubscript{ui}=uid\textsubscript{BC}(bc), me:BCMet\textsubscript{L} me=mere\textsubscript{L}(bc),
256d btt:Bus\textsubscript{TimTbl} btt=attr\textsubscript{Bus\textsubscript{TimTbl}}(bc) }
256a ||
256e || { bus\textsubscript{bui}(b\textsubscript{ui},me)(ln,btt,bpos)
256e b:B b ∈ bs, b\textsubscript{ui}:B\textsubscript{UI} b\textsubscript{ui}=uid\textsubscript{B}(b), me:BMet\textsubscript{L} me=mere\textsubscript{L}(b), ln:LN pln=attr\textsubscript{LN}(b),
256e btt:Bus\textsubscript{TimTbl} btt=attr\textsubscript{Bus\textsubscript{TimTbl}}(b), bpos:BPos bpos=attr\textsubscript{BPos}(b) }
256a ||
256f || { automobile\textsubscript{aui}(a\textsubscript{ui},me,rn)(apos)
256f a:A a ∈ as, a\textsubscript{ui}:A\textsubscript{UI} a\textsubscript{ui}=uid\textsubscript{A}(a), me:AMet\textsubscript{L} me=mere\textsubscript{L}(a),
256f rn:Reg\textsubscript{No} r\textsubscript{no}=attr\textsubscript{Reg\textsubscript{No}}(a), apos:APos apos=attr\textsubscript{APos}(a) }
CHAPTER 8. Bibliography
Bibliography


www.imm.dtu.dk/~db/todai/tse-1.pdf,
Univ. of Denmark, Fredsvej 11, DK-2840 Holte, Denmark, 2010.

Report 2013-2, DTU Compute and Fredsvej 11, DK-2840 Holte, 
Denmark, Spring 2013.

[15] Dines Bjørner. A Rôle for Mereology in Domain Science and 
Engineering. Synthese Library (eds. Claudio Calosi and Pierluigi 
Graziani). Springer, Amsterdam, The Netherlands, October 
2014.

Experimental Research, Technical University of Denmark, 
Fredsvej 11, DK-2840 Holte, Denmark, November 2016.


DK-2840 Holte, Denmark, September 2018. An incomplete draft report; currently 60+ pages.


[34] Patrick Cousot and Rhadia Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *4th POPL*: 


viewpoint of eternity]. Munksgaard · Rosinante, Copenhagen, Denmark, 1997. 200 pages.


[64] Kai Sørlander. Indføring i Filosofien [Introduction to The Philosophy]. Informations Forlag, Copenhagen, Denmark, 2016. 233 pages.


