Domain Analysis & Description – Sorts, Types, Intents

Submission for the Klaus Havelund Festschrift, 25 October 2020

Dines Bjørner

Technical University of Denmark, DK-2800 Kgs.Lyngby, Denmark
Fredsvej 11, DK-2840 Holte, Danmark
bjorner@gmail.com, www.imm.dtu.dk/~db

Abstract

In earlier publications on **domain analysis & description** [1–5, 8] we introduced the notion of discrete endurants, both natural and artefactual, being parts and characterised classes of these as **sorts**. Parts were then analysed with respect to internal qualities such as unique identifiers, mereologies and attributes and these were characterised in terms of **types**. In [7] we show how Kai Sørlander's philosophy [14–16] justifies our ontology of entities not on empirical grounds, but on philosophical grounds – and we brought forward the notion of **intentional pull** mentioned only briefly in [8]. In [6] we further analysed certain attribute types in terms of the *SI: The International System of Units*[1]. In this paper we shall examine some aspects of sorts, types and intents not covered in [1–8].[2]

1 Introduction

By a **domain** we shall understand a **rationally describable** segment of a **human assisted** reality, i.e., of the world, its **physical parts: natural** ["God-given"] and **artifactual** ["man-made"], and **living species**: **plants** and **animals** including, notably, **humans**. These are **endurants** ("still"), as well as **perdurants** ("alive"). Emphasis is placed on **"human-assistedness"**, that is, that there is *at least one (man-made)* **artifact** and, therefore, that **humans** are a primary cause for change of endurant **states** as well as perdurant **behaviours**.

1.1 Entities, Endurants and Perdurants

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* [13, Vol. I, pg. 665] ■ **Examples:** A train, a train ride, an aircraft, a flight ■

Endurant By an **endurant** 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"* [13, Vol. I, pg. 656]. Were we to "freeze" time we would still be able to observe the entire endurant ■ **Examples:** A road, an automobile, a human driver ■

Perdurant By a **perdurant** 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 perdurant, alternatively an entity is perdurant if it endures continuously, over time, persists, lasting [13, Vol. II, pg. 1552] ■ **Examples:** A train ride, an aircraft flight ■

[1] https://en.wikipedia.org/wiki/International_System_of_Units
[2] September 27, 2019: 15:26

1.2 Discrete and Continuous Endurants

Discrete Endurant By a **discrete endurant** we shall understand an endurant which is separate, individual or distinct in form or concept ■ **Examples:** A pipeline and its individual units: pipes, valves, pumps, forks, etc. ■

Continuous Endurants: Non-solids By a **continuous endurant** (a **non-solid**) we shall understand an endurant which is prolonged, without interruption, in an unbroken series or pattern ■ **Examples:** Water, oil, gas, compressed air, etc. A container, which we consider a discrete endurant, may contain a non-solid, like a gas pipeline unit may contain gas ■

1.3 A Domain Ontology

Figure 1 graphs an essence of the domain ontology of entities, endurants, perdurants, etc., as these concepts were covered in [8]. Sections 1.1 – 1.2 covered some aspects of the first three layers, from the top, of that domain ontology. Following [8], as also justified, on grounds of philosophy, by [7], we shall claim that the

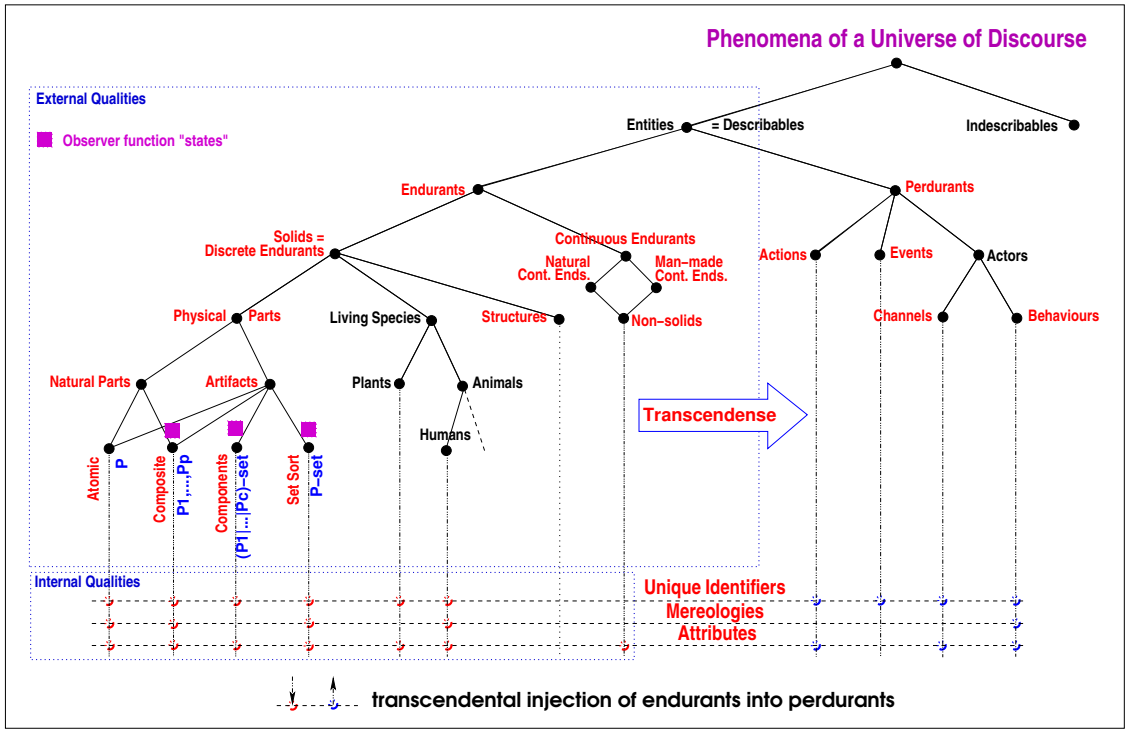

Fig. 1. A Domain Ontology

manifest world, i.e., the *physical* and *living endurants*, can be analysed with respect to their observable, i.e., viewable and touchable, i.e., **external qualities**, respectively their measurable, i.e., **internal qualities**. The external qualities are summarised in **sorts**. Values of sorts, i.e., *physical* and *living endurants* [we shall omit treatment of *structures* in this paper], can be summarised in three (*internal quality*) categories: *unique identifiers*, *mereologies*, and *attributes*. These *internal qualities* are summarised by **types**[3].

[3] The **RAISE** [12] **S**pecification **L**anguage. **RSL** [11], as we use it in this paper, does not distinguish between *sorts* and *types*.

2 Sorts, Types, Intents

We shall, in this paper, make a oragmatics distinction between sorts and types. Sorts will be used to characterise observable endurants. Types will be used to characterise sorts ! Intents are then [something] associated with man-made endurants.

2.1 Sorts

By a **sort** we shall generally mean a named set of values which we do not, at the instance of introducing the sort name, further define. By a **domain analysis & description sort** we shall in this paper specifically mean a named sort of endurants. In this paper we shall use the term 'sort' in this later sense.

Physical Parts, Living Species and Structures With discrete endurants we associate sorts.

Physical Parts: By a *physical part* we shall understand a discrete endurant existing in time and subject to laws of physics, including the *causality principle* and *gravitational pull*[4] ■ Classes of "similar" physical parts are given names and these we shall refer to as sort names. Our investigation into sorts, types and intents will focus on physical, in particular artefactual parts.

Living Species: By a *living species* we shall understand a discrete 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*; which they must be *causally determined to maintain*. This *development and maintenance* must further 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*; another form which, **additionally**, can be characterised by the *ability to purposeful movement* The first we call **plants**, the second we call **animals** ■ We shall not, in this paper further deal with living species

Structures: By a **structure** we shall understand a discrete endurant which the domain engineer chooses to describe as consisting of one or more endurants, whether discrete or continuous, but to **not** endow with ***internal qualities***: unique identifiers, mereology or attributes ■ We shall not, in this paper further deal with the concept of structures.

Natural Parts and Artefacts Physical parts are either *natural parts*, or are *artefacts*, i.e. man-made parts, which possess **internal qualities**: **unique identification**, **mereology**, and one or more **attributes** ■ For more on internal qualities, see Sect. 2.2.

Natural Parts: Natural parts are in *space* and *time*; are subject to the *laws of physics*, and also subject to the *principle of causality* and *gravitational pull* ■ **Examples:** an island, a mountain, a river, a lake, a granite rock, a gold lode ■

Artefacts: By an **artifact** we shall understand a *man-made physical part* **Examples:** road nets, road intersections (**hub**s), **link**s (roads between adjacent hubs); automobiles ■

[4] This characterisation is the result of our study of relations between philosophy and computing science, notably influenced by Kai Sørlander's Philosophy. We refer to our research report [7].

Various Forms of Physical Parts We now arrive at the point where **sorts** come into play. Natural parts are either **atomic**, or **composite**, and artefactual parts are either of **atomic** sort, or of **composite** sort, or of **set sort**, or of **components** sort.

Atomic Parts are those which, in a given context, are deemed to *not* consist of meaningful, separately observable proper *sub-part*s. A **sub-part** is a *part* ■ **Examples:** a hub, a link, a pipe, a valve, a wheel, an engine, a door, a window ■

Composite Parts are those which, in a given context, are deemed to *indeed* consist of meaningful, separately observable proper *sub-part*s ■ **Examples:** an automobile, a road net, a pipeline ■

Components come in sets. That is, sets of sets of components of two or more distinct sorts. In general the domain analyser cum describer chooses to **not** endow components with ***mereology*** ■ **Examples:** A postal letter box may contain letters, small parcels, newspapers and advertisement brochures ■

Set Sort Parts are simplifications of components. A set sort part is a set of parts of the *same* sort. The domain analyser cum describer chooses to **indeed** endow components with **mereology** ■ **Examples:** Road nets are considered compositions of two parts. a hub aggregate and a link aggregate. The hub aggregate is a set sort part and consists of a set of hubs; the link aggregate is a set sort part and consists of a set of links ■ Component and set sort parts are pragmatic constructions.

Analysis and Description Prompts Implicit in the "story" of Sect. 2.1 are the following **analysis prompts:**

- is_entity
- is_endurant
- is_perdurant
- is_discrete
- is_continuous
- is_phys._part
- is_liv._species
- is_structure
- is_natural_part
- is_artefact
- is_atomic
- is_composite ■
- is_components ■
- is_set_sort ■
- et cetera (■)

The ■ boxes imply analysis states where the following **description prompts** are applicable:

- observe_composite_sorts
- observe_component_sorts
- observe_set_sort

respectively (– et cetera).

The description observers can be formalised, for example:

type: observe_composite_sorts: E→**Text**

Narrative:
s. narrative text on sorts $E_1,...,E_n$
o. narrative text on observers obs_E_1,...,obs_E_n
p. narrative text on proof obligation: $\mathscr{P}$
Formalisation:
s. **type** $E_1,...,E_n$
o. **value** obs_E_1: E→E_1,...,obs_E_n: E→E_n
p. **proof obligation** $\mathscr{P}$: $\forall\ i{:}\{1..n\}\bullet$is_$E_i$(e)≡$\bigwedge\{\sim E_j(e)|j{:}[1..n]\setminus\{i\}|j{:}[1..n]\}$

type: observe_set_sort: E→**Text**

Narratives:
s. narrative text on sort P
o. narrative text on observer obs_Ps
Formalisation:
s. **type** P, Ps = P-**set**
o. **value** obs_Ps: E→P-**set**

For further details see [8]. It is high time for an example.

An Example, I of III: Road Transport

External Qualities

1 The road transport system consists of two aggregates: a road net and automobiles.
2 The road net consists of aggregates of atomic hubs (street intersections) and atomic links (streets).
3 Hub aggregates are sets of hubs and link aggregates are sets of links.
4 Automobile aggregates are sets of automobiles.

type
1. RTS, RN, AA
value
1. obs_RN: RTS → RN
1. obs_AA: RTS → AA
type
2. AH, AL
value
2. obs_AH: RN → AH
2. obs_AL: RN → AL

type
3. Hs = H-**set**, H
3. Ls = L-**set**, L
value
3. obs_Hs: AH → Hs
3. obs_Ls: AL → Ls
type
4. As = A-**set**, A
value
4. obs_As: AA → As

2.2 Types

By a **type** we shall generally mean a named set of values which we, at the instance of introducing the type name, either define as an atomic **token** type, or as a *concrete* type. By an atomic token type we mean a set of further undefined atomic values. By a concrete type we shall here mean either a **set** of **T** typed values, i.e., **T-set**, or a **list** of **T** typed values, i.e., $\mathbf{T}^*$, or a **map** from **A** typed values to **B** typed values, i.e., $\mathbf{A} \overset{m}{\rightarrow} \mathbf{B}$, or a **Cartesian product** (a "record", a "structure") of **A, B, ..., C** typed values, i.e., $\mathbf{A} \times \mathbf{B} \times \cdots \times \mathbf{C}$. A type can also be a **union** type, that is, the set union of distinct types **A, B, ..., C**, i.e., $\mathbf{A} | \mathbf{B} | \cdots | \mathbf{C}$. **Token**s, **Integer**s, **Natural Number**s, **Real**s and **Character**s are base types. [Concrete types of common programming languages include **array**s and **record**s.]

Space and Time

Space: There is an abstract notion of (definite) $\mathbb{SPACE}$(s) of further unanalysable points; and there is a notion of $\mathbb{POINT}$s in $\mathbb{SPACE}$. Space is not an attribute of endurants. Space is just there. So we do not define an observer, `observe_space`.

5 A point observer, observe_$\mathbb{POINT}$, is a function which applies to a[ny] specific "location" on a physical endurant, e, and yields a point, $\ell : \mathbb{POINT}$.

value
5 obs_$\mathbb{POINT}$: E → $\mathbb{POINT}$

Time: By a **definite time** we shall understand an abstract representation of time such as for example year, month, day, hour, minute, second, et cetera ■We shall not be concerned with any representation of time. That is, we leave it to the domain analyser cum describer to choose an own representation [10]. Similarly we shall not be concerned with any representation of time intervals.[5]

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

6 So there is an abstract type $\mathbb{T}ime$,
7 and an abstract type $\mathbb{TI}$: $\mathbb{T}ime\mathbb{I}nterval$.
8 There is no $\mathbb{T}ime$ origin, but there is a "zero" $\mathbb{TI}$me interval.
9 One can add (subtract) a time interval to (from) a time and obtain a time.
10 One can add and subtract two time intervals and obtain a time interval – with subtraction respecting that the subtrahend is smaller than or equal to the minuend.
11 One can subtract a time from another time obtaining a time interval respecting that the subtrahend is smaller than or equal to the minuend.
12 One can multiply a time interval with a real and obtain a time interval.
13 One can compare two times and two time intervals.

```
type
6  𝕋
7  𝕋𝕀
value
8  0:𝕋𝕀
9  +,−: 𝕋 × 𝕋𝕀 → 𝕋
10 +,−: 𝕋𝕀 × 𝕋𝕀 ~→ 𝕋𝕀
11 −: 𝕋 × 𝕋 → 𝕋𝕀
12 ∗: 𝕋𝕀 × Real → 𝕋𝕀
13 <,≤,=,≠,≥,>: 𝕋 × 𝕋 → Bool
13 <,≤,=,≠,≥,>: 𝕋𝕀 × 𝕋𝕀 → Bool
axiom
9 ∀ t:𝕋 • t+0 = t
```

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

```
value
14 record_𝕋𝕀𝕄𝔼(): Unit → 𝕋
```

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

Internal Qualities The internal qualities of endurants may include: unique identifiers, for physical parts and living species; mereologies, for atomic, composite, set sort and human parts; and attributes, for physical parts and living species.

Unique Identifiers Every discrete endurant, e:E, is unique and can hence be ascribed a **unique identifier**; that identifier can be ascertained by applying the **uid_E** observer function to e.

Mereologies Mereology is the study of parts and the wholes they form ■ We shall interpret the **mereology of a part**, p, here as as the topological and/or conceptual relations between that part and other parts. Typically we can express the mereology of p, i.e., **mereo_P**(p), in terms of the sets of unique identifiers of the other parts with which p is related. Generally, we can express that relationship as a triplet: **mereo_P**(p)=(ips,iops,ops) where ips is the set of unique identifiers of those parts "from" which p "receives input", whatever 'input' means (!); iops is the set of unique identifiers of those parts "with" which p mutually "shares" properties, whatever 'shares' means (!); ops is the set of unique identifiers of those parts "to" which p "delivers output", whatever 'output' means (!); and where the three sets are mutually disjoint.

Attributes Part attributes form more "free-wheeling" sets of **internal qualities** than those of unique identifiers and mereologies.

Parts and non-solids are typically recognised because of their spatial form and are otherwise characterised by their intangible, but measurable attributes. That is, whereas endurants, whether discrete (as are parts and components) or continuous (as are materials), are physical, tangible, in the sense of being spatial [or being abstractions, i.e., concepts, of spatial endurants], attributes are intangible: cannot normally be touched, but can be objectively measured. Thus, in our quest for describing domains where humans play an active rôle, we rule out subjective "attributes": feelings, sentiments, moods. Thus we shall abstain, in our domain science also from matters of aesthetics.

Thus, to any part and non-solid, e, we can associate one or more attributes A_1, A_2, ..., A_m, where A_i is an attribute type name and where **attr_**A_i(e) is the corresponding attibute observer.

Internal Quality Observers We can summarise the observers for internal qualities while otherwise referring to [8] for details.

type observe_unique_identifier: P→**Text**

Narratives:
i. text on unique identifier: UI
o. text on unique identifier observer: uid_E
Formalisation:
i. **type** UI
o. **value** uid_E: E → UI

type observe_mereology: P→**Text**

Narratives:
m. text on mereology: M
o. text on mereology observer: mereo_E
Formalisation:
m. **type** M = $\mathscr{E}$(UI$_a$, ..., UI$_c$)
o. **value** mereo_E: E → M

In the expression of $\mathscr{E}$(UI$_a$,...,UI$_c$) the domain analyser cum describer need not take into consideration any concern for possible data structure efficiency as we are not prescribing software requirments let alone specifying a software design.

type observe_attributes: P→**Text**

Narratives:
a. texts on attributes: A$_i$, ..., A$_k$
o. texts on attribute observers: attr_A$_i$, ..., attr_A$_k$
Formalisation:
a. **type** A$_i$ [= $\mathscr{A}_i$], ..., A$_k$ [= $\mathscr{A}_k$]
o. **value** obs_A$_i$: E → A$_i$, ..., obs_A$_k$: E → A$_k$

where [= $\mathscr{A}_j$] refer to an optional type expression.

In the expression of $\mathscr{A}_j$ the domain analyser cum describer need not take into consideration any concern for possible data structure efficiency as we are not prescribing software requirments let alone specifying a software design.

An Example, I of III: Road Transport

Internal Qualities

We shall only be concerned with the internal qualities of hubs, links and automobiles. First unique identifiers and mereologies.

15 Hubs, links and automobiles have unique indentifiers.
16 The mereology of hubs is a pair: finite set of zero, one or more link identifiers and a set of (the) automobile identifiers (allowed to traverse the hubs).
17 The mereology of links is a a pair: a two element set of distinct hub identifiers and a set of (the) automobile identifiers (allowed to traverse the links).
18 The mereology of automobiles is the set of hub and link identifiers of (the) hubs and links (the automobiles are allowed to traverse).

type
15. HI, LI, AI

```
value
15.  uid_H:H→HI, uid_L:L→LI, uid_A:A→AI
type
16.  HM = LI-set × AI-set
17.  LM = HI-set × AI-set axiom ∀ (his,_):LM•card his=2
18.  AM = (HI|LI)-set
value
16.  mereo_H: H → HM
17.  mereo_L: L → LM
18.  mereo_A: A → AM
```

We omit description of detailed wellformedness conditions such as: the hubs and links that automobiles may traverse must be duly noted in automobile mereologies, and exactly those; the hubs and links that automobiles may traverse must form a connected road net; et cetera. Then attributes.

19 Hub attributes:
   a number of lanes, surface, etc.;
   b state: set of pairs of link identifiers from, respectively to which automabiles may traverse the hub;
   c state space: set of all possible hub states;
   d traversal history: the recording of which autombiles traversed the hub at which time.

```
type
19a.  NoL, SUR, ...
value
19a.  attr_NoL:H→NoL,attr_SUR:H→SUR,...
type
19b.  HΣ = (LI×LI)-set
19c.  HΩ = HΣ-set
19d.  HHist = AI ~m~> T*
value
19b.  attr_HΣ:H→HΣ
19c.  attr_HΩ:H→HΩ
19d.  attr_HHist:H→HHist
```

We omit expression of wellformedness conditions.

20 Link attributes:
   a number of lanes, surface etc.;
   b state: set of 0, 1, 2 or 3 pairs of adjacent hub identifiers, the link is closed, open in one direction (closed in the opposite), open in the other direction, or open in both directions;
   c state space: set of all possible link states;
   d traversal history: the recording of which autombiles traversed the link at which link position and at which time.

```
type
20a.  NoL, SUR, ...
value
20a.  attr_NoL:L→NoL,attr_SUR:L→SUR,...
type
20b.  LΣ = (LI×LI)-set
20c.  LΩ = HΣ-set
20d.  LHist = AI ~m~> (T×LPos)*
21b.  LPos = ...
value
20b.  attr_LΣ:L→LΣ
20c.  attr_LΩ:L→LΩ
```

20d. attr_LHist:L→LHist

21 Automobile attributes:
  a Owner, license number, ...
  b Vehicle position
  c Automobile history: time-ordered sequence of recordings of vehicle positions.

type
21a. OWN, LIC, ...
21b. VPos == HPos | LPos
21b. HPos :: HI
21b. LPos :: LI × (HI×Frac×HI)
21c. AHist = (𝕋×VPos)*

Physics Attributes Typically, when physicists write computer programs, intended for calculating physics behaviours, they "lump" all of these into the **type Real**, thereby hiding some important pysics 'dimensions'. In this section we shall review that which is missing !

Example I of III: Specific Attributes

In **Example II of III: Internal Qualities** we did not mention such obvious automobile attributes as velocity, say in kilometers/hour, and acceleration, say in meters/second^2. We shall now consider that problem !

MORE TO COME

SI: The International System of Quantities In physics we operate on values of attributes of manifest, i.e., physical phenomena. The type of some of these attributes are recorded in well known tables, cf. Tables 1–3.

SI Units Table 1 shows the base units of physics.

Base quantity	Name	Type
length	meter	m
mass	kilogram	kg
time	second	s
electric current	ampere	A
thermodynamic temperature	kelvin	K
amount of substance	mole	mol
luminous intensity	candela	cd

Table 1. Base SI Units

Derived Units Table 2 on the following page shows the units of physics derived from the base units.

Further Units Table 3 on the next page shows further units of physics derived from the base units.

Standard Prefixes for SI Units of Measure Table 4 on Page 11 shows standard prefixes for SI units of measure.
Table 5 on Page 11 shows fractions of SI units of measure.

Name	Type	Derived Quantity	Derived Type
radian	rad	angle	m/m
steradian	sr	solid angle	$m^2 \times m^{-2}$
Hertz	Hz	frequency	s^{-1}
newton	N	force, weight	$kg \times m \times s^{-2}$
pascal	Pa	pressure, stress	N/m^2
joule	J	energy, work, heat	$N \times m$
watt	W	power, radiant flux	J/s
coulomb	C	electric charge	$s \times A$
volt	V	electromotive force	W/A ($kg \times m^2 \times s^{-3} \times A^{-1}$)
farad	F	capacitance	C/V ($kg^{-1} \times m^{-2} \times s^4 \times A^2$)
ohm	Ω	electrical resistance	V/A ($kg \times m^2 \times s^3 \times A^2$)
siemens	S	electrical conductance	A/V ($kg1 \times m^2 \times s^3 \times A^2$)
weber	Wb	magnetic flux	$V \times s$ ($kg \times m^2 \times s^{-2} \times A^{-1}$)
tesla	T	magnetic flux density	Wb/m^2 ($kg \times s^2 \times A^{-1}$)
henry	H	inductance	Wb/A ($kg \times m^2 \times s^{-2} \times A^2$)
degree Celsius	^{o}C	temp. rel. to 273.15 K	K
lumen	lm	luminous flux	$cd \times sr$ (cd)
lux	lx	illuminance	lm/m^2 ($m^2 \times cd$)

Table 2. Derived SI Units

Name	Explanation	Derived Type
area	square meter	m^2
volume	cubic meter	m^3
speed, velocity	meter per second	m/s
acceleration	meter per second squared	m/s^2
wave number	reciprocal meter	m-1
mass density	kilogram per cubic meter	kg/m^3
specific volume	cubic meter per kilogram	m3/kg
current density	ampere per square meter	A/m^2
magnetic field strength	ampere per meter	A/m
substance concentration	mole per cubic meter	mol/m3
luminance	candela per square meter	cd/m^2
mass fraction	kilogram per kilogram	kg/kg = 1

Table 3. Further SI Units

To give a hint at what we are aiming at, let us consider the following possibility. In describing manifest, primarily discrete domains with, however, some dynamics. An example could be that of the road transport example.

The point in bringing this material is that when modelling, i.e., describing domains we must be extremely careful in not falling into the trap of modelling physics types, etc., as we do in programming – by simple **Real**s. We claim, without evidence, that many trivial programming mistakes are due to confusions between especially derived SI units, fractions and prefixes.

MORE TO COME

Domain Oriented Programming Languages: One could, rather easily, augment standard programming languages, for use in physics calculations, to feature a refined type system that reflects the SI units, simple and composite, as well as standard SI prefixes and fractions. In the early 00's a student of mine, as his MSc. thesis, designed a set of domain specific programing languages, one for high school physics students, one for business college selling/purchase/warehouse applications, et cetera. We refer to the very elegant domain-specific actuarial programming language, **Actulus**, [9] for life insurance and pensions. Our

Prefix name		deca	hecto	kilo	mega	giga
Prefix symbol		da	h	k	M	G
Factor	10^0	10^1	10^2	10^3	10^6	10^9
Prefix name		tera	peta	exa	zetta	yotta
Prefix symbol		T	P	E	Z	Y
Factor		10^{12}	10^{15}	10^{18}	10^{21}	10^{24}

Table 4. Standard Prefixes for SI Units of Measure

Prefix name		deci	centi	milli	micro	nano
Prefix symbol		d	c	m	μ	n
Factor	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-6}	10^{-9}
Prefix name		pico	femto	atto	zepto	yocto
Prefix symbol		p	f	a	z	y
Factor		10^{-12}	10^{-15}	10^{-18}	10^{-21}	10^{-24}

Table 5. Fractions

Domain Specific Language, **DSL**, dogma is this: the design (and semantics) of any DSL must be based on a carefully analysed and both informally and formally described domain.

Summary

MORE TO COME

2.3 Intents

Artefacts are made with an **intent**: one or more purposes for which the parts are to serve. **Examples:** roads are "made to accomodate" automobiles, and automobiles are "made to drive" on roads ■

We do not here suggest a formal way of expressing intents. That is, we do not formalise "made to accomodate", "made to drive", et cetera ! Intents, instead, are expressed as **intentional pulls** !

Examples of Intents

TO BE WRITTEN

Intentional Pull The term "intentional pull" was first introduced in [8].

MORE TO COME

Summary

MORE TO COME

2.4 Summary

3 Conclusion

4 References

[1] Dines Bjørner. Domain Engineering. In Paul Boca and Jonathan Bowen, editors, *Formal Methods: State of the Art and New Directions*, Eds. Paul Boca and Jonathan Bowen, pages 1–42, London, UK, 2010. Springer.

[2] Dines Bjørner. Domain Science & Engineering – *From Computer Science to The Sciences of Informatics, Part I of II: The Engineering Part*. *Kibernetika i sistemny analiz*, 2(4):100–116, May 2010.

[3] Dines Bjørner. Domain Science & Engineering – *From Computer Science to The Sciences of Informatics Part II of II: The Science Part*. *Kibernetika i sistemny analiz*, 2(3):100–120, June 2011.

[4] Dines Bjørner. Manifest Domains: Analysis & Description. *Formal Aspects of Computing*, 29(2):175–225, March 2017. Online: 26 July 2016.

[5] Dines Bjørner. Domain Facets: Analysis & Description. Technical report, Technical University of Denmark, Fredsvej 11, DK-2840 Holte, Denmark, May 2018. Extensive revision of [1]. imm.dtu.-dk/~dibj/2016/facets/faoc-facets.pdf.

[6] Dines Bjørner. The Manifest Domain Analysis & Description Approach to Implicit and Explicit Semantics. *EPTCS: Electronic Proceedings in Theoretical Computer Science, Yasmine Ait-Majeur, Paul J. Gibson and Dominique Méry*, 12 May 2018. First International Workshop on Handling IMPlicit and EXplicit Knowledge in Formal Fystem Development, 17 November 2017, Xi'an, China. imm.dtu.dk/~dibj/2017/bjorner-impex.pdf.

[7] Dines Bjørner. Domain Analysis & Description – A Philosophy Basis. Technical report, Technical University of Denmark, Fredsvej 11, DK-2840 Holte, Denmark, August 2019. imm.dtu.-dk/~dibj/2019/filo/main2.pdf.

[8] Dines Bjørner. Domain Analysis & Description – Principles, Techniques and Modelling Languages. *ACM Trans. on Software Engineering and Methodology*, 29(2):..., April 2019. imm.dtu.-dk/~dibj/2018/tosem/Bjorner-TOSEM.pdf.

[9] David R. Christiansen, Klaus Grue, Henning Niss, Peter Sestoft, and Kristján S. Sigtryggsson. Actulus Modeling Language - An actuarial programming language for life insurance and pensions. Technical Report, http://www.edlund.dk/sites/default/files/Downloads/paper_actulus-modeling-language.pdf, Edlund A/S, Denmark, Bjerregårds Sidevej 4, DK-2500 Valby. (+45) 36 15 06 30. edlund@edlund.dk, http://www.edlund.dk/en/insights/scientific-papers, 2015. This paper illustrates how the design of pension and life insurance products, and their administration, reserve calculations, and audit, can be based on a common formal notation. The notation is human-readable and machine-processable, and specialised to the actuarial domain, achieving great expressive power combined with ease of use and safety.

[10] Carlo A. Furia, Dino Mandrioli, Angelo Morzenti, and Matteo Rossi. *Modeling Time in Computing*. Monographs in Theoretical Computer Science. Springer, 2012.

[11] Chris W. George, Peter Haff, Klaus Havelund, Anne Elisabeth Haxthausen, Robert Milne, Claus Bendix Nielsen, Søren Prehn, and Kim Ritter Wagner. *The RAISE Specification Language*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1992.

[12] Chris W. George, Anne Elisabeth Haxthausen, Steven Hughes, Robert Milne, Søren Prehn, and Jan Storbank Pedersen. *The RAISE Development Method*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1995.

[13] W. Little, H.W. Fowler, J. Coulson, and C.T. Onions. *The Shorter Oxford English Dictionary on Historical Principles*. Clarendon Press, Oxford, England, 1973, 1987. Two vols.

[14] Kai Sørlander. *Det Uomgængelige – Filosofiske Deduktioner [The Inevitable – Philosophical Deductions, with a foreword by Georg Henrik von Wright]*. Munksgaard · Rosinante, 1994. 168 pages.

[15] Kai Sørlander. *Under Evighedens Synsvinkel [Under the viewpoint of eternity]*. Munksgaard · Rosinante, 1997. 200 pages.
[16] Kai Sørlander. *Indføring i Filosofien [Introduction to The Philosophy]*. Informations Forlag, 2016. 233 pages.

