Domain Analysis & Description – Sorts, Types, Intents

DINES BJØRNER, TECHNICAL UNIVERSITY OF DENMARK

ACM Reference Format:
Dines Bjørner, Technical University of Denmark. . Domain Analysis & Description – Sorts, Types, Intents. 1, 1 (November), 9 pages.

ABSTRACT

In earlier publications on **domain analysis & description** [5–8, 10, 12] 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 [11] we show how Kai Sørlander's philosophy [24–26] 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 [12]. In [9] 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 [5–12].[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 **artefactual** ["man-made"], and **living species**: **plants** and **animals** including, notably, **humans**. These entities are **endurants** ("still"), as well as **perdurants** ("alive"). Emphasis is placed on **"human-assistedness"**, that is, 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

1.1.1 **Entity:** By an **entity** we shall understand a **phenomenon**, i.e., something that can be *observed*: 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* [23, Vol. I, pg. 665] ■ **Examples:** A train, a train ride, an aircraft, a flight ■

1.1.2 **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"* [23, 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 ■

1.1.3 **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 [23, Vol. II, pg. 1552] ■ **Examples:** A train ride, an aircraft flight ■

1.2 Discrete and Continuous Endurants

1.2.1 **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. ■

1.2.2 **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 [12]. Sections 1.1 – 1.2 covered some aspects of the first three layers, from the top, of that domain ontology. Following [12], as also justified,

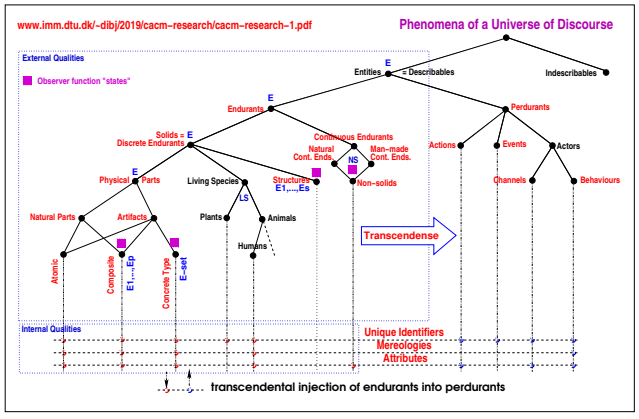

Fig. 1. A Domain Ontology

on grounds of philosophy, by [11], we shall claim that the manifest

[1] https://en.wikipedia.org/wiki/International_System_of_Units
[2] November 21, 2019: 15:51

Dines Bjørner, Technical University of Denmark.

XXXX-XXXX//11-ART $15.00 doi.org/

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].

•••

We shall, in this paper, make a pragmatic 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 SORTS

By a **sort** we shall generally mean a named set of endurants which we shall later further characterise.

2.1 Physical Parts, Living Species and Structures

With discrete endurants we associate sorts.

2.1.1 **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.

2.1.2 **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

2.1.3 **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.

2.2 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. 3.2.

[3]The **RAISE** [19] **S**pecification **L**anguage. **RSL** [18], as we use it in this paper, does not distinguish between *sorts* and *types*.

[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 [11].

2.2.1 **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 ■

2.2.2 **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 ■

2.3 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**.

2.3.1 **Atomic Parts:** **Atomic Part**s 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 ■

2.3.2 **Composite Parts:** **Composite Part**s 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 ■

2.3.3 **Set Sort Parts:** **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 ■ Set sort parts are pragmatic constructions.

2.4 Analysis and Description Prompts

Implicit in the "story" of Sects. 2.1–2.3 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:

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** $\mathscr{P}$: ∀ i:{1..n}•is_E_i(e)≡⋀{∼E_j(e)|j:[1..n]\{i}|j:[1..n]}

In any specific domain analysis & description the analyser cum describer chooses which subset of composite sorts to analyse & describe. That is: any one domain model emphasises certain aspects and leaves out many "other" aspects.

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**

Typically P may be a sort expression: P1|P2|...|Pn where Pi are sorts.

2.5 An Example: 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

3 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 values of type **T**, i.e., **T-set**, or a **list** of values of type **T**, i.e., **T***, or a **map** from values of type **A** to values of type **B**, i.e., **A** $\overrightarrow{m}$ **B**, or a **Cartesian product** (a "record", a "structure") of **A, B, ..., C** typed values, i.e., **A**×**B**×⋯×**C**. A type can also be a **union** type, that is, the set union of distinct types **A, B, ..., C**, i.e., **A**|**B**|⋯|**C**. **Token**s, **Integer**s, **Natural Number**s, **Real**s, **Character**s, $\mathbb{POINT}$, $\mathbb{T}$, and $\mathbb{TI}$, for the latter three, see Sect. 3.1, are base, or "atomic", types. Concrete types of common programming languages include **array**s (**vector**s, **matrice**s, **tensor**s, etc.) and **record**s. Eventually it all ends up in atomic (i.e., base) types.

3.1 Space and Time

Space and time "fall" somewhat outside a "standard view" of types. We do not prescribe, really, a type **space**. It is just there. We shall present a view of time different from those of [14, 16, 27].

3.1.1 **Space:** There is an abstract notion of (definite) $\mathbb{SPACE}$(s) of further un-analysable 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, ℓ : $\mathbb{POINT}$.

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

3.1.2 **Time:** By a **definite time** we shall understand an abstract representation of time such as for example year, 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 [16]. Similarly we shall not be concerned with any representation of time intervals.[5]

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 $\mathbb{T}$
7 $\mathbb{TI}$
value
8 **0**:$\mathbb{TI}$
9 +,−: $\mathbb{T}$ × $\mathbb{TI}$ → $\mathbb{T}$
10 +,−: $\mathbb{TI}$ × $\mathbb{TI}$ $\xrightarrow{\sim}$ $\mathbb{TI}$
11 −: $\mathbb{T}$ × $\mathbb{T}$ → $\mathbb{TI}$
12 ∗: $\mathbb{TI}$ × **Real** → $\mathbb{TI}$
13 <,≤,=,≠,≥,>: $\mathbb{T}$ × $\mathbb{T}$ → **Bool**
13 <,≤,=,≠,≥,>: $\mathbb{TI}$ × $\mathbb{TI}$ → **Bool**
axiom
9 ∀ t:$\mathbb{T}$ • t+**0** = t

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

value
14 **record**_$\mathbb{TIME}$(): **Unit** → $\mathbb{T}$

[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.

The time recorder applies to nothing and yields a time. **record_TIME()** can only occur in action, event and behavioural descriptions.

3.2 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.

3.2.1 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.

3.2.2 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.

3.2.3 Attributes:

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

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 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 attribute observer.

3.2.4 Internal Quality Observers:

We can summarise the observers for internal qualities while otherwise referring to [12] 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

For 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 requirements let alone specifying a software design. The choice of $\mathscr{E}$(UI$_a$,...,UI$_c$), that is, of the mereology of any one sort $\mathscr{E}$, depends on the aspects of the domain that its analyser cum describer wishes to study. That is, "one and the same domain" may give rise to different models each emphasizing their aspects.

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 requirements let alone specifying a software design.

One and "seemingly" the same domain may give rise to different analyses & descriptions. Each of these emphasize different aspects. **Example: Road Net:** In one model of a road net emphasis may be on automobile traffic (aiming, eventually, at a road pricing system). I another model of "the same" road net emphasis may be on the topological layout (aiming, eventually, at its construction). In yet a third model "over" a road net emphasis may be on traffic control ■ For each such "road net" model the domain analyser cum describer selects different overlapping sets of attributes.

3.2.5 Three Categories of Attributes:

We can identify three kinds of attributes: (i) physics, (i) artefactual and (i) intentional.

3.3 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 physics 'dimensions'. In this section we shall review that which is missing !

The subject of physical dimensions in programming languages is rather decisively treated in David Kennedy's 1996 PhD Thesis [22] — so there really is no point in trying to cast new light on this subject other than to remind the reader of what these physical dimensions are all about.

3.3.1 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. Table 1 on the next page 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

Table 2 on the facing page shows the units of physics derived from the base units. Table 3 shows further units of physics derived from

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×m
watt	W	power, radiant flux	J/s
coulomb	C	electric charge	s×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×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	°C	temp. rel. to 273.15 K	K
lumen	lm	luminous flux	cd×sr (cd)
lux	lx	illuminance	lm/m^2 ($m^2 \times cd$)

Table 2. Derived SI Units

the base units. The upper half of Table 5 shows standard prefixes for

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

SI units of measure and the lower half of Table 5 shows fractions of SI units.

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

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.

Units are Atomic. A volt, $kg \times m^2 \times s^{-3} \times A^{-1}$, see Table 2, is atomic. It is not a composite structure of mass, length, time, and electric current – in some intricate relationship.

Example 1: Physics Attributes

Hub attributes:

15 number of lanes, surface, etc.;

type
15. NoL, SUR, ...

value
15. attr_NoL:H→NoL
15. attr_SUR:H→SUR, ...

Link attributes:

16 number of lanes, surface. etc.

value
16. attr_NoL:L→NoL
16. attr_SUR:L→SUR, ...

Automobile attributes:

17 Power, Fuel (Gasoline, Diesel, Electric, ...), Size, ...

18 Velocity, Acceleration, ...

type
17. BHp = **Nat**:**kg**×**m**$^{-2}$×**s**$^{-3}$
17. Fuel
17. Length = **Nat**:**cm**
17. Width = **Nat**:**cm**
17. Height = **Nat**:**cm**
18. Vel = **Real**:**m**×**s**$^{-1}$
18. Acc = **Real**:**m**×**s**$^{-2}$

value
17. attr_BHp: A→BHp
17. attr_Fuel: A→Fuel
17. attr_Length: A→Length
17. attr_Width: A→Width
17. attr_Height: A→Height
18. attr_Vel: A→Vel
18. attr_Acc: A→Acc

•••

Physical attributes may ascribe mass and volume to endurants. But they do not reveal the substance, i.e., the material from which the endurant is made. That is done by chemical attributes.

3.3.2 Chemical Elements:

The *mole*, mol, substance is about chemical molecules. A mole contains exactly $6.02214076 \times 10^{23}$ (the Avogadro number) constituent particles, usually atoms, molecules, or ions – of the elements, cf. 'The Periodic Table', en.wikipedia.org/wiki/Periodic_table, cf. Fig. 2. Any specific molecule is then a compound of two or more elements, for example, calcium-phosphat: Ca3(PO4)2.

Moles bring substance to endurants. The physics attributes may ascribe weight and volume to endurants, but they do not explain what it is that gives weight, i.e., fills out the volume.

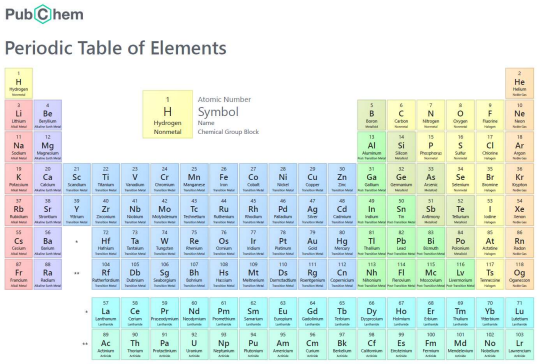

Fig. 2. Periodic Table

3.4 Artefactual Attributes

3.4.1 Examples of Artefactual Attributes:

We exemplify some artefactual attributes.

- **Designs.** Artefacts are man-made endurants. Hence "exhibit" a design. My three dimensional villa has floor plans, etc. The artefact attribute: *'design'* can thus be presented by the architect's or the construction engineer's CAD/CAM drawings.
- **States** of an artefact, such as, for example, a road intersection (or railway track) traffic signal; and
- **Currency**, e.g., Kr, $, £, €, ¥, et cetera, used as an attribute[6], say the cost of a train ticket.
- **Artefactual Dimensions.** Let the domain be that of industrial production whose attributes could then be: production: units produced per year, Units/Year; growth: increase in units produced per year, Units×Year^{-2}; productivity: production per staff, Units×Year^{-1}×Staff^{-1} — where the base for units and staff are natural numbers.

Document Artefactual Attributes

Let us consider *documents* as artefactual parts. Typical document attributes are: (i) kind of document: `book, report, pamphlet, letter` and `ticket`, (ii) publication date, (iii) number of pages, (iv) author/publisher and (v) possible colophon information. *All of these attributes are non-physics quantities.*

Road Net Artefactual Attributes

Hub attributes:

19 state: set of pairs of link identifiers from, respectively to which automobiles may traverse the hub;
20 state space: set of all possible hub states.

type	value
19. HΣ = (LI×LI)-**set**	19. attr_HΣ:H→HΣ
20. HΩ = HΣ-**set**	20. attr_HΩ:H→HΩ

Link attributes:

21 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; and
22 state space: set of all possible link states.

type	value
21. LΣ = (LI×LI)-**set**	21. attr_LΣ:L→LΣ
22. LΩ = LΣ-**set**	22. attr_LΩ:L→LΩ

[6]One could also consider a [10 €] bank note to be an artefact, i.e., a part.

4 INTENTS

4.1 Expressing Intents

Artefacts are made with an **intent**: one or more purposes for which the parts are to serve. Usually intents involve two or more part sorts.

Examples of Intents.

- **Road Transport:** roads are "made to accommodate" automobiles, and automobiles are "made to drive" on roads ■
- **Credit Card System:** credit cards are for "payment of purchased merchandise", and retailers are "there to sell merchandise" ■

4.2 Intent Modelling

We do not here suggest a formal way of expressing intents. That is, we do not formalise "made to accommodate", "made to drive", et cetera ! Intents, instead, are expressed as **intentional pulls**, and these are then expressed in terms of "intent-related" attributes.

Examples of Intent-related Attributes. The intent-related attributes are not based on physical evidence, but on what we can, but do not necessarily speak about.

Example: Intentional Attributes

Road Transport:

23 Hub traversal history: the recording of which automobiles traversed a hub at which time.
24 Link traversal history: the recording of which automobiles traversed a link at which time.
25 Automobile history: the recording of which hubs and links were traversed at which time.

type	value
23. HHist=AI $\xrightarrow{m}$ℙ-**set**	23. attr_HHist:H→HHist
24. LHist=AI $\xrightarrow{m}$ℙ-**set**	24. attr_LHist:L→LHist
25. AHist=(HI|LI) $\xrightarrow{m}$ℙ-**set**	25. attr_AHist:A→AHist

All three history attributes are subject to constraints: the automobile, hub and link identifiers must be of automobiles, hubs and links of a (i.e., the) road net; the same automobile cannot be at two or more hubs and/or links at any one time (23–24) and the timed visits must be commensurate with the road net; et cetera.

Credit Card System:

26 Credit card histories X "records" Y[7], by time, the shop and the merchandise bought.
27 Shop histories "record", by time, the credit card and the merchandise sold.

type
26. CHist = $\mathbb{T}$ $\overrightarrow{m}$ (SI×MI)
27. SHist = $\mathbb{T}$ $\overrightarrow{m}$ (CI×MI)

value
26. attr_CHist: C → CHist
27. attr_SHist: S → SHist

The two history attributes are subject to constraints: the shop and credit card identifiers must be of shops and credit cards of the credit card system; and the merchandise identifier must be of a merchandise of the identified shop.

4.3 Intentional Pull

The term 'intentional pull' was first introduced in [12].

An Aside: Road Transport System States. In order to express intentional pulls we need introduce a notion of states. In general, by a state, we shall mean any collection of parts each of which contains one or more dynamic attributes, that is, attributes whose values may change.

Road Net States &c.

We shall consider the following states:

28 any road transport system;
29 the set of all its hubs;
30 the set of all its links; and
31 the set of all its automobiles.

value
28. rtn:RTN
29. hs:obs_Hs(obs_AH(rtn))
30. ls:obs_Ls(obs_AL(rtn))
31. as:obs_As(obs_AA(rtn))

32 From the set of hubs we can extract the map of hub histories: from hub identifiers to hub histories.

type
32. HHists=HI $\overrightarrow{m}$(AI $\overrightarrow{m}$$\mathbb{T}$-**set**)
value
32. extr_HHists:H-**set**→HHists
32. extr_HHists(hs) ≡ [hi↦attr_HHist(h)|h:H•h ∈ hs ∧ hi=uid_H(h)]

33 From the set of links we can extract the map of link histories: from link identifiers to link histories.

type
33. LHists=LI $\overrightarrow{m}$(AI $\overrightarrow{m}$$\mathbb{T}$-**set**)
value
33. extr_LHists:L-**set**→LHists
33. extr_LHists(ls) ≡ [li↦attr_LHist(l)|l:L•l ∈ ls ∧ li=uid_L(l)]

34 From the set of automobiles we can extract the map of automobile histories: from automobile identifiers to automobile histories.

type
34. AHists=AI $\overrightarrow{m}$((HI|LI) $\overrightarrow{m}$$\mathbb{T}$-**set**)
value
34. extr_AHists:A-**set**→AHists
34. extr_AHists(as) ≡ [ai↦attr_AHist(a)|a:A•a ∈as ∧ ai=uid_A(a)]

35 We can merge the hub and link histories.

type
35. HLHists=(HI|LI) $\overrightarrow{m}$(AI $\overrightarrow{m}$$\mathbb{T}$-**set**)
value
35. mergeHLHists:(HHists×LHists)→HLHists
35. mergeHLHists(hhists,lhists) ≡ hhists ∪ lhists

36 The ahists:AI $\overrightarrow{m}$((HI|LI) $\overrightarrow{m}$$\mathbb{T}$-**set**) can be "inverted": inv_ahists(ahists)[8] into hlhists:(HI|LI) $\overrightarrow{m}$(AI $\overrightarrow{m}$$\mathbb{T}$-**set**)
37 and then "re-inverted": inv_hlhists(hlhists) into ahists′:AI $\overrightarrow{m}$((HI|LI) $\overrightarrow{m}$$\mathbb{T}$-**set**)
38 to [re-]obtain ahists as no automobile can be in any two or more places at any one time.

value
36. inv_ahists: AHists → HLHists
36. inv_ahists(ahists) ≡
36. [hli↦[ai′↦(ahists(ai′))(hli)|ai′:AI•ai′∈ **dom** ahists∧hli∈ **dom** ahists(ai′)]
36. |ai:AI,hli:(HI|LI)•ai∈ **dom** ahists∧hli∈ **dom** ahists(ai)]
assertion:
38. ∀ ahists:AHists • inv_hlhists(inv_ahists(ahists)) = ahists
where:
37. inv_hlhists: HLHists → AHists
37. inv_hlhists(hlhists) ≡ left to the reader

Examples of Intentional Pulls. Road Transport:

39 If an automobile history records that an automobile was at a hub or on a link at some time,
then that hub, respectively link, history records that that automobile was there at that time,
and vice versa — and only that.

intentional pull:
39. □ ∀ rtn:RTN, hs:Hs, ls:Ls, as:As•
39. hs=obs_Hs(obs_AH(rtn))
39. ∧ ls=obs_Ls(obs_AL(rtn))
39. ∧ as=obs_As(obs_AA(rtn))
39. ∧ **let** ahists=xtr_AHists(as),
39. hlhists=xtr_HHists(hs)∪xtr_LHists(ls) **in**
39. inv_AHists(ahists) = hlhists **end**

Credit Card System:

40 If a credit card history records that an purchase was made at a shop of some merchandise and at
41 some time,
42 then that shop's history records that that such a purchase was made there at that time,
43 and vice versa — and only that.

We leave the formalisation to the reader ■

5 ACTIONS, EVENTS, BEHAVIOURS

By a **transcendental deduction** [12] we shall interpret discrete endurants as behaviours. Behaviours are sets of sequences of actions, events and behaviours. Behaviours communicate, for example, by means of CSP channels and output/input commands [21].

5.1 Actions

Actions are functions which purposefully are initiated by behaviours and potentially changes a state. Actions apply to behaviour arguments and yield updates to these.

5.2 Events

Events are functions which surreptitiously "occur" to behaviours, typically instigated by "some outside", and usually changes a state. Events updates behaviour arguments. Events can be expressed as CSP [21] inputs.

5.3 Behaviours

To every part we shall, in principle, associate a behaviour. The behaviour is unique identified by the unique identifier of the part. The behaviour communicates with such other parts as are identified by

[8] Note the subtle use of free and bound variables in the map comprehension expressions.

the mereology of the parts. The behaviour otherwise depends on arguments: the unique part identifier, the part mereology, the part attributes separated into the static attributes, i.e., those with constant values, the programmable attributes, and the remaining dynamic attributes. The programmable attributes are those whose values are set by the behaviour, i.e., its actions.

5.4 Summary

The "miracle" of transcendental deduction is fundamental to domain analysis & description. It "ties" sorts: their external and internal qualities strongly to the dynamics of domains. Details on transcendental deductions, actions, events and behaviours are given in [12].

6 CONCLUSION

The sort, type and intent concepts of the domain analysis & description method covered in [12] has been studied in further detail. Although, as also illustrated by Fig. 1 on Page 1, the method includes the analysis of natural and living species, it is primarily aimed at artefacts and domains dominated by such. We refer to [13] for a dozen or so examples of medium-scale domain analysis & description case studies. You will see from those examples that they are all rather frugal with respect to ascribing attributes. That is: An endurant may have very many attributes, but in any one domain description in which it is present the analyser cum describer may have chosen to "abstract some out (!)", that is, to not consider some — often very many of these — of these attributes.

6.1 Sort versus Types

"Sorts are not recursive !" That is, parts of sort *S* do not contain proper sub-parts of same sort *S*.

6.1.1 **Pragmatics:** In this paper we have used the terms 'sorts' and 'types' as follows. **Sorts** are used to describe external qualities of endurants: whether discrete or continuous (solids or non-solids), whether physical parts, living species or structures, whether natural parts or artefacts, and whether atomic, composite, components or set sorts. **Types** are used to describe internal qualities of endurants: unique identifiers, mereologies, and attributes.

6.1.2 **Syntactics: Sorts** are defined by simple identifiers:

- **type** S.

Types are defined either by base type definitions **type** T = BTE, where BTE is an atomic type expression, for example either of,

- **Intg**[:Dim],
- **Nat**[:Dim],
- **Real**[:Dim],
- **Char**[:Dim],
- **Token**,
- $\mathbb{POINT}$ and
- $\mathbb{T}, \mathbb{TI}$.

where [:Dim] is either absent or some standard prefix and fraction SI unit. Or types are defined by composite type expressions, **type** T = CTE, for example of the form:

CTE = A-**set** | B×C×...×D | E $\overrightarrow{m}$ F | etc.

where A, B, C, ..., D, E, F, etc., are type expressions – where [recursive] T is allowed.

6.1.3 **Semantics:** We start with types. **Types** are sets of either base (type) values, or structures over these: sets of sets (of etc.), sets of Cartesians (of etc.), sets of maps (from etc.), et cetera. **Sorts** are sets of endurants as characterised by their being discrete or continuous (solids or non-solids), physical parts, living species or structures, natural parts or artefacts, and atomic, composite, components or set sorts; and as furthermore characterised by the types of their possible unique identifiers, possible mereologies (components have no mereologies), and attributes.

6.2 An Earlier Review of Types

Section 5.3, Pages 43–48, of [8], brings an extensive review of published papers on types. That review does not make the distinction made in this paper as summarised in Sect. 6.1.

MORE TO COME

6.3 Domain Oriented Programming Languages

I found out about Kennedy's work from [20]. My own interest in the subject goes back to the early 1980s. Around year 2000 I had an MSc student work out formal specifications and compilers for two "small" programming languages: one for senior high school [student] physics and one for business college [student] accounting. I otherwise refer to [2, Exercise 9.4, Page 235].

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.

We refer to the very elegant domain-specific actuarial programming language, **Actulus**, [15] for life insurance and pensions.

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

6.4 Research Topics

- **Artefactual Types:** A further study of artefactual types seems reasonable: *are there identifiable categories of artefactual types ?*
- **Intents:** We have remarked that we suggest no formal representation of intents. But should there be ?
- **Intentional Pull:** Although we have illustrated some *"intentional pulls"*, also in [11, 12], it seems only reasonable to study further examples.

Attributes of Living Species: The Swedish botanist, zoologist, and physician, *Carl von Linné*, is the father of modern taxonomy: the science that finds, describes, classifies, and names living things, published [28, in 1748]. In domain analysing & describing living species one, of course, cannot really contribute much new. So we leave that area to the living species taxonomists – while referring to [17, *Formal Concept Analysis — Mathematical Foundations*]. See also [8, Sect. 1.8].

6.5 Acknowledgements

It is a pleasure to acknowledge my collaboration over recent years with Klaus Havelund.

[11]

REFERENCES

[1] Dines Bjørner. Domain Models of "The Market" — in Preparation for E–Transaction Systems. In *Practical Foundations of Business and System Specifications (Eds.: Haim Kilov and Ken Baclawski)*, The Netherlands, December 2002. Kluwer Academic Press. http://www2.imm.dtu.dk/~dibj/themarket.pdf.

[2] Dines Bjørner. *Software Engineering, Vol. 2: Specification of Systems and Languages*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006. Chapters 12–14 are primarily authored by Christian Krog Madsen. See [3, 4].

[3] Dines Bjørner. *Software Engineering, Vol. 2: Specification of Systems and Languages*. Qinghua University Press, 2008.

[4] Dines Bjørner. **Chinese:** *Software Engineering, Vol. 2: Specification of Systems and Languages*. Qinghua University Press. Translated by Dr Liu Bo Chao et al., 2010.

[5] 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.

[6] 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.

[7] 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.

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

[9] Dines Bjørner. Domain analysis & description - the implicit and explicit semantics problem. In Régine Laleau, Dominique Méry, Shin Nakajima, and Elena Troubitsyna, editors, Proceedings Joint Workshop on *Handling IMPlicit and EXplicit knowledge in formal system development (IMPEX)* and *Formal and Model-Driven Techniques for Developing Trustworthy Systems (FM&MDD)*, Xi'An, China, 16th November 2017, volume 271 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–23. Open Publishing Association, 2018.

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

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

[12] 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.

[13] Dines Bjørner. Domain Case Studies:
- 2019: *Container Line*, ECNU, Shanghai, China imm.dtu.dk/~db/container-paper.pdf
- 2018: *Documents*, TongJi Univ., Shanghai, China imm.dtu.dk/~dibj/2017/docs/docs.pdf
- 2017: *Urban Planning*, TongJi Univ., Shanghai, China imm.dtu.dk/~dibj2017/up/urban-planning.pdf
- 2017: *Swarms of Drones*, Inst. of Softw., Chinese Acad. of Sci., Peking, China imm.dtu.dk/~dibj/2017/swarms/swarm-paper.pdf
- 2013: *Road Transport*, Techn. Univ. of Denmark imm.dtu.dk/~dibj/road-p.pdf
- 2012: *Credit Cards*, Uppsala, Sweden imm.dtu.dk/~dibj/2016/credit/accs.pdf
- 2912: *Weather Information*, Bergen, Norway imm.dtu.dk/~dibj/-2016/wis/wis-p.pdf
- 2010: *Web-based Transaction Processing*, Techn. Univ. of Vienna, Austria imm.dtu.dk/~dibj/wfdftp.pdf
- 2010: *The Tokyo Stock Exchange*, Tokyo Univ., Japan imm.dtu.dk/~db/todai/tse-1.pdf, imm.dtu.dk/~db/todai/tse-2.pdf
- 2009: *Pipelines*, Techn. Univ. of Graz, Austria imm.dtu.dk/~dibj/pipe-p.pdf
- 2007: *A Container Line Industry Domain*, Techn. Univ. of Denmark imm.dtu.dk/~dibj/container-paper.pdf
- 2002: *The Market*, Techn. Univ. of Denmark imm.dtu.dk/~dibj/themarket.pdf, [1]
- 1995–2004: *Railways*, Techn. Univ. of Denmark - a compendium imm.dtu.dk/~dibj/train-book.pdf

Experimental research reports, Technical University of Denmark, Fredsvej 11, DK-2840 Holte, Denmark.

[14] Wayne D. Blizard. A Formal Theory of Objects, Space and Time. *The Journal of Symbolic Logic*, 55(1):74–89, March 1990.

[15] 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, 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.

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

[17] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis — Mathematical Foundations*. Springer-Verlag, January 1999.

[18] 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.

[19] 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.

[20] J Paul Gibson and Dominique Méry. Explicit modelling of physical measures: From event-b to java. In Régine Laleau, Dominique Méry, Shin Nakajima, and Elena Troubitsyna, editors, Proceedings Joint Workshop on *Handling IMPlicit and EXplicit knowledge in formal system development (IMPEX)* and *Formal and Model-Driven Techniques for Developing Trustworthy Systems (FM&MDD)*, Xi'An, China, 16th November 2017, volume 271 of *Electronic Proceedings in Theoretical Computer Science*, pages 64–79. Open Publishing Association, 2018.

[21] Charles Anthony Richard Hoare. *Communicating Sequential Processes*. C.A.R. Hoare Series in Computer Science. Prentice-Hall International, 1985. Published electronically: usingcsp.com/cspbook.pdf (2004).

[22] Andrew Kennedy. *Programming languages and dimensions*. PhD thesis, University of Cambridge, Computer Laboratory, April 1996. 149 pages: cl.cam.ac.uk/techreports/UCAM-CL-TR-391.pdf. Technical report UCAM-CL-TR-391, ISSN 1476-298.

[23] 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.

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

[25] Kai Sørlander. *Under Evighedens Synsvinkel [Under the viewpoint of eternity]*. Munksgaard · Rosinante, 1997. 200 pages.

[26] Kai Sørlander. *Indføring i Filosofien [Introduction to The Philosophy]*. Informations Forlag, 2016. 233 pages.

[27] Johan van Benthem. *The Logic of Time*, volume 156 of *Synthese Library: Studies in Epistemology, Logic, Methhodology, and Philosophy of Science (Editor: Jaakko Hintika)*. Kluwer Academic Publishers, P.O.Box 17, NL 3300 AA Dordrecht, The Netherlands, second edition, 1983, 1991.

[28] Carl von Linné. *An Introduction to the Science of Botany*. Lipsiae [Leipzig]: Impensis Godofr. Kiesewetteri, 1748.