Dines Bjørner's MAP-i Lecture # 1

An Overview Of Domain Description

Monday, 25 May 2015: 10:30–11:15

1. Domain Analysis & Description

Abstract

- We show that manifest domains,
  - an understanding of which are
  - a prerequisite for software requirements prescriptions,

  can be precisely described:

  - narrated and
  - formalised.

- We show that manifest domains can be understood as a collection of
  - endurant, that is, basically spatial entities:
    - parts,
    - components and
    - materials,

  and
  - perdurant, that is, basically temporal entities:
    - actions,
    - events
    - and behaviours.

- We show that parts can be modeled in terms of
  - external qualities whether:
    - atomic or
    - composite

    parts,
- having internal qualities:
  - unique identifications,
  - mereologies, which model relations between parts, and
  - attributes.

- We show the manifest domain analysis endeavour can be supported by a calculus of manifest domain analysis prompts:

- `is_entity`,
- `is_endurant`,
- `is_perdurant`,
- `is_part`,
- `is_component`,
- `is_material`,
- `is_atomic`,
- `is_composite`,
- `has_components`,
- `has_materials`,
- `has_concrete_type`,
- `attribute_names`,
- `is_stationary`, etcetera.

- We show how the manifest domain description endeavour can be supported by a calculus of manifest domain description prompts:
  - `observe_part_sorts`,
  - `observe_part_type`,
  - `observe_components`,
  - `observe_materials`,
  - `observe_unique_identifier`,
  - `observe_mereology`,
  - `observe_attributes`,
  - `observe_location` and
  - `observe_position`.

- We show how to model essential aspects of perdurants in terms of their signatures based on the concepts of endurants.
- And we show how one can “compile”
  - descriptions of endurant parts into
  - descriptions of perdurant behaviours.
- We do not show prompt calculi for perdurants.
- The above contributions express a method
  - with principles, technique and tools
  - for constructing domain descriptions.

1.1. Introduction

- The broader subject of this seminar is that of software development.
- The narrower subject is that of manifest domain engineering.
- We see software development
  in the context of the `TripTych` approach.

- The contribution of this seminar is twofold:
  - ⟐ the propagation of manifest domain engineering
    - ⊚ as a first phase of the development of
    - ⊚ a large class of software —

    and
    - ⊚ a set of principles, techniques and tools
    - ⊚ for the engineering of the analysis & descriptions
    - ⊚ of manifest domains.

- These principles, techniques and tools are embodied in a set of analysis and description prompts.
  - We claim that this embodiment in the form of prompts is novel,
  - that the (yet to be investigated) "calculus" is a first such "method calculus".

1.1.1. The TripTych Approach to Software Engineering

- We suggest a **TripTych** view of software engineering:
  - *before software can be designed and coded*
  - *we must have a reasonable grasp of "its" requirements;*
  - *before requirements can be prescribed*
  - *we must have a reasonable grasp of "the underlying" domain.*

- To us, therefore, software engineering contains the three sub-disciplines:
  - domain engineering,
  - requirements engineering and
  - software design.

- This seminar contributes, we claim, to a **methodology** for **domain analysis** &[1] **domain description**.
- References [dines:ugo65:2008]
  - show how to "refine" **domain descriptions** into **requirements prescriptions**,

  and reference [DomainsSimulatorsDemos2011]
  - indicates more general relations between **domain descriptions** and
    - **domain demos**,
    - **domain simulators** and
    - more general **domain specific software**.

[1] When, as here, we write A & B we mean A & B to be one subject.

- In branches of engineering based on natural sciences
  - professional engineers are educated in these sciences.
  - Telecommunications engineers know Maxwell's Laws.
    - Maybe they cannot themselves "discover" such laws,
    - but they can "refine" them into designs,
    - for example, for mobile telephony radio transmission towers.
  - Aeronautical engineers know laws of fluid mechanics.
    - Maybe they cannot themselves "discover" such laws,
    - but they can "refine" them into designs,
    - for example, for the design of airplane wings.
  - And so forth for other engineering branches.

- Our point is here the following:
  - software engineers must domain specialise.
  - This is already done, to a degree, for designers of
    - compilers,
    - operating systems,
    - database systems,
    - Internet/Web systems,

    etcetera.
  - But is it done for software engineering
    - banking systems,
    - traffic systems,
    - health care,
    - insurance, etc. ?
  - We do not think so, but we claim it should be done.

1.1.2. Method and Methodology

1.1.2.1. Method

- By a **method** we shall understand
  - a "somehow structured" set of `principles`
  - for `select`ing and `apply`ing
  - a number of `techniques` and `tools`
  - for `analysing` problems and `synthesizing` solutions
  - for a given domain ■[2]

[2]Definitions and examples are delimited by ■ respectively ■ symbols.

- The 'somehow structuring' amounts,
  - in this treatise on domain analysis & description,
  - to the techniques and tools being related to a set of
  - domain analysis & description "prompts",
  - "issued by the method",
  - prompting the domain engineer,
  - hence carried out by the domain analyser & describer[3] —
  - conditional upon the result of other prompts.

[3] We shall thus use the term domain engineer to cover both the analyser & the describer.

1.1.2.2. Discussion

- There may be other 'definitions' of the term 'method'.
- The above is the one that will be adhered to in this seminar.
- The main idea is that
  - there is a clear understanding of what we mean by, as here,
    - a software development method,
    - in particular a *domain analysis & description method.*

- The **main principles** of the `TripTych` domain analysis and description approach are those of
  - abstraction and both
    - narrative and
    - formal
  - modeling.
  - This means that evolving domain descriptions
    - necessarily limit themselves to a subset of the domain
    - focusing on what is considered relevant, that is,
    - abstract "away" some domain phenomena.

- The **main techniques** of the `TripTych` domain analysis and description approach are
  - besides those techniques which are in general associated with formal descriptions,
  - focus on the techniques that relate to the deployment of of the individual prompts.

- And the **main tools** of the `TripTych` domain analysis and description approach are
  - the analysis and description prompts and the
  - description language, here the `R`aise `S`pecification `L`anguage `RSL`.

- A main contribution of this seminar is therefore
  - that of "painstakingly" elucidating the
    - principles,
    - techniques and
    - tools

    of the domain analysis & description method.

1.1.2.3. Methodology

- By **methodology** we shall understand
  - the study and knowledge
  - about one or more methods[4] ■

[4]Please note our distinction between method and methodology. We often find the two, to us, separate terms used interchangeably.

1.1.3. Computer and Computing Science

- By **computer science** we shall understand
  - the study and knowledge of
    - the conceptual phenomena
    - that "exists" inside computers
  - and, in a wider context than just computers and computing,
    - of the theories "behind" their
    - formal description languages ■
- Computer science is often also referred to as theoretical computer science.

- By **computing science** we shall understand
  - the study and knowledge of
    - how to construct
    - and describe

    those phenomena ■
- Another term for computing science is programming methodology.

- This paper is a computing science paper.
  - It is concerned with the construction of domain descriptions.
  - It puts forward a calculus for analysing and describing domains.
  - It does not theorize about this calculus.
  - There are no theorems about this calculus and hence no proofs.
  - We leave that to another study and paper.

1.1.4. What Is a Manifest Domain ?

- We offer a number of complementary delineations of what we mean by a manifest domain.
- But first some examples, "by name" !

Example 1 . Manifest Domain Names: Examples of suggestive names of manifest domains are:

- *air traffic,*
- *banks,*
- *container lines,*
- *documents,*
- *hospitals,*
- *pipelines,*
- *railways* and
- *road nets* ■

- A **manifest domain** is a
  - human- and
  - artifact-assisted
  - arrangement of
    - **endurant**, that is spatially “stable”, and
    - **perdurant**, that is temporally “fleeting”

    entities.
  - Endurant entities are
    - either parts
    - or components
    - or materials.
  - Perdurant entities are
    - either actions
    - or events
    - or behaviours

Example 2 . Manifest Domain Endurants: Examples of (names of) endurants are

- **Air traffic:** *aircraft, airport, air lane.*
- **Banks**: *client, passbook.*
- **Container lines:** *container, container vessel, terminal port.*
- **Documents:** *document, document collection.*
- **Hospitals:** *patient, medical staff, ward, bed, medical journal.*
- **Pipelines:** *well, pump, pipe, valve, sink, oil.*
- **Railways:** *simple rail unit, point, crossover, line, track, station.*
- **Road nets:** *link (street segment), hub (street intersection)* ■

Example 3 . Manifest Domain Perdurants: Examples of (names of) perdurants are

- **Air traffic:** *start (ascend) an aircraft, change aircraft course.*
- **Banks:** *open, deposit into, withdraw from, close (an account).*
- **Container lines:** *move container off or on board a vessel.*
- **Documents:** *open, edit, copy, shred.*
- **Hospitals:** *admit, diagnose, treat (patients).*
- **Pipelines:** *start pump, stop pump, open valve, close valve.*
- **Railways:** *switch rail point, start train.*
- **Road nets:** *set a hub signal, sense a vehicle* ■

- A **manifest domain** is further seen as a mapping
  - from *entities*
  - to *qualities*,

  that is, a mapping
  - from manifest phenomena
  - to usually non-manifest qualities ■

Example 4 . Endurant Entity Qualities: Examples of (names of) endurant qualities:

- **Pipeline:**
  - *unique identity of a pipeline unit,*
  - *mereology (connectedness) of a pipeline unit,*
  - *length of a pipe,*
  - *(pumping) height of a pump,*
  - *open/close status of a valve.*
- **Road net:**
  - *unique identity of a road unit (hub or link),*
  - *road unit mereology:*
    - *identity of neighbouring hubs of a link,*
    - *identity of links emanating from a hub,*
  - *and state of hub (traversal) signal* ■

Example 5 . Perdurant Entity Qualities: Examples of (names of) perdurant qualities:

- **Pipeline:**
  - *the signature of an open (or close) valve action,*
  - *the signature of a start (or stop) pump action,*
  - *etc.*
- **Road net:**
  - *the signature of an insert (or remove) link action,*
  - *the signature of an insert (or remove) hub action,*
  - *the signature of a vehicle behaviour,*
  - *etc.* ■

- Our definitions of what a manifest domain is
  - are, to our own taste, not fully adequate;
  - they ought be so sharp that one can unequivocally distinguish such domains that are not manifest domains from those which are (!).
  - Examples of the former are:
    - the Internet,
    - language compilers,
    - operating systems,
    - data bases,

    etcetera.
- As we progress we shall sharpen our definition of ‘manifest domain’.

We shall in the rest of this seminar just write ‘domain’ instead of ‘manifest domain’.

1.1.5. What Is a Domain Description ?

- By a **domain description** we understand
  - a collection of pairs of
  - narrative and commensurate
  - formal

  texts, where each pair describes
  - either aspects of an endurant entity
  - or aspects of a perdurant entity ■

- What does it mean that some text describes a domain entity ?
- For a text to be a **description text** it must be possible
  - to either, if it is a narrative,
    - to reason, informally, that the ***designated*** entity
    - is described to have some properties
    - that the reader of the text can observe
    - that the described entities also have;
  - or, if it is a formalisation
    - to prove, mathematically,
    - that the formal text
    - ***denotes*** the postulated properties ■

Example 6 . Narrative Description of Bank System Endurants:

1 A banking system consists of a bank and collections of clients and of passbooks.

2 A bank attribute is that of a general ledger.

3 A collection of clients is a set of uniquely identified clients.

4 A collection of passbooks is a set of uniquely identified passbooks.

5 A client "possess" zero, one or more passbook identifiers.

6 Two or more clients may share the same passbook.

7 The general ledger records, for each passbook identifier, amongst others, the set of one or more client identifiers sharing that passbook, etc.

Etcetera ■

Example 7 . Formal Description of Bank System Endurants:

type
1. B, CC, CPB

value
1. **obs_part**_CC: B → CC,
1. **obs_part**_CPB: B → CPB

type
2. GL

value
2. **attr**_GL: B → GL

type
3. C, CI, CC = C-**set**,
4. PB, PBI, CPB = PB-**set**

value
5. **attr**_C: C → PBI-**set**

type
7. GL = PBI $\overrightarrow{m}$ SH × ...
7. SH = PBI-**set**

Etcetera ■

Example 8 . Narrative Description of Bank System Perdurants:

8 Clients and the bank possess cash (i.e., monies).

9 Clients can open a bank account and receive in return a passbook.

10 Clients may deposit monies into an account in response to which the passbook and the general ledger are updated.

11 Clients may withdraw monies from an account: if the balance of monies in the designated account is not less than the requested amount the client is given the (natural number) designated monies and the passbook and the general ledger are updated.

Etcetera ■

Example 9 . Formal Description of Bank System Perdurants:

type

8. M

value

8. **attr**_M: (B|C) $\rightarrow$ M
9. open: B $\rightarrow$ B $\times$ PB
10. deposit: PB $\rightarrow$ M $\rightarrow$ B $\rightarrow$ B $\times$ PB
11. withdraw: PB $\rightarrow$ B $\rightarrow$ **Nat** $\stackrel{\sim}{\rightarrow}$ B $\times$ PB $\times$ M

Etcetera ■

- By a **domain description** we shall thus understand a text which describes
  - the **entities** of the domain:
    - whether **endurant** or **perdurant**,
    - and when endurant whether
      - **discrete** or **continuous**,
      - **atomic** or **composite**;
    - or when perdurant whether
      - **action**s,
      - **event**s or
      - **behaviour**s.
  - as well as the **qualities** of these **entities**.

- So the task of the domain analyser cum describer is clear:
  - There is a domain: right in front of our very eyes,
  - and it is expected that that domain be described.

1.1.6. Towards a Methodology of Domain Analysis & Description

1.1.6.0.1 Practicalities of Domain Analysis & Description

- How does one go about analysing & describing a domain ?
  - Well, for the first,
    - one has to designate one or more **domain analysers** cum
    - **domain describers**,
    - i.e., trained **domain scientists** cum **domain engineers**.
  - How does one get hold of a **domain engineer** ?
    - One takes a **software engineer** and *educates* and *trains* that person in
      - **domain science** &
      - **domain engineering**.
    - A derivative purpose of this seminar is to unveil aspects of **domain science** & **domain engineering**.

- The education and training consists in bringing forth
  - a number of scientific and engineering issues
    - of **domain analysis** and
    - of **domain description**.
  - Among the engineering issues are such as:
    - *what do I do when confronted*
      - *with the task of domain analysis?* and
      - *with the task of description?* and
    - *when, where and how do I*
      - `select` *and* `apply`
      - *which* `techniques` *and which* `tools`?

- Finally, there is the issue of
  - *how do I, as a domain describer, choose appropriate*
    - *abstractions and*
    - *models ?*

1.1.6.0.2 The Four Domain Analysis & Description "Players"

- We can say that there are four 'players' at work here.
  - the domain,
  - the domain analyser & describer,
  - the domain analysis & description method, and
  - the evolving domain analysis & description.

- The ***domain*** is there.
  - The domain analyser & describer cannot change the domain.
  - Analysing & describing the domain does not change it[5].
  - In a meta-physical sense it is inert.
  - In the physical sense the domain will usually contain
    - entities that are static (i.e., constant), and
    - entities that are dynamic (i.e., variable).

[5] Observing domains, such as we are trying to encircle the concept of domain, is not like observing the physical world at the level of subatomic particles. The experimental physicists' instruments of observation changes what is being observed.

- The ***domain analyser & domain describer*** is a human,
  - preferably a scientist/engineer[6],
  - well-educated and trained in domain science & engineering.
  - The domain analyser & describer
    - observes the domain,
    - analyses it according to a method and
    - thereby produces a domain description.

[6] At the present time domain analysis appears to be partly an art, partly a scientific endeavour. Until such a time when domain analysis & description principles, techniques and tools have matured it will remain so.

- As a concept the ***method*** is here considered "fixed".
  - By 'fixed' we mean that its principles, techniques and tools do not change during a domain analysis & description.
  - The domain analyser & describer
    - may very well apply these principles, techniques and tools
    - more-or-less haphazardly,
    - flaunting the method,
    - but the method remains invariant.
  - The method, however, may vary
    - from one domain analysis & description (project)
    - to another domain analysis & description (project).
  - Domain analysers & describers do become wiser from a project to the next.

- Finally there is the evolving *domain analysis & description*.
  - That description is a text, usually both informal and formal.
  - Applying a *domain description prompt* to the domain
    - yields an *additional domain description text*
    - which is added to the thus evolving *domain description*.

- One may speculate of the rôle of the "input" domain description.
  - Does it change?
  - Does it help determine the additional domain description text?
  - Etcetera.
- Without loss of generality we can assume
  - that the "input" domain description is changed and
  - that it helps determine the added text.

- Of course, analysis & description is a trial-and-error, iterative process.
  - During a sequence of analyses,
  - that is, analysis prompts,
  - the analyser "discovers"
  - either more pleasing abstractions
  - or that earlier analyses or descriptions
  - were wrong.
  - So they are corrected.

1.1.6.0.3 An Interactive Domain Analysis & Description Dialogue

- We see domain analysis & description
  - as a process involving the above-mentioned four 'players',
  - that is, as a dialogue
  - between the domain analyser & describer and the domain,
  - where the dialogue is guided by the method
  - and the result is the description.
- We see the method as a 'player' which issues prompts:
  - alternating between:
  - ***"analyse this"*** (analysis prompts) and
  - ***"describe that"*** (synthesis or, rather, description prompts).

1.1.6.0.4 Prompts

- In this paper we shall suggest
  - a number of *domain analysis prompts* and
  - a number of *domain description prompts*.
- The **domain analysis prompt**s,
  - (schematically: `analyse_named_condition(e)`)
  - directs the analyser to inquire
  - as to the truth of whatever the prompt "names"
  - at wherever part (component or material), `e`, in the domain the prompt so designates.

- Based on the truth value of an analysed entity the domain analyser may then be prompted to describe that part (or material).
- The **domain description prompts**,
  - (schematically: `describe_type_or_quality(e)`)
  - directs the (analyser cum) describer to formulate
  - both an informal and a formal description
  - of the type or qualities of the entity designated by the prompt.
- The prompts form languages, and there are thus two languages at play here.

1.1.6.0.5 A Domain Analysis & Description Language

- The 'Domain Analysis & Description Language' thus consists of a number of meta-functions, the prompts.
  - The meta-functions have names (say `is_endurant`) and types,
  - but have no formal definition.
  - They are not computable.
  - They are "performed"
    by the domain analysers & describers.
  - These meta-functions are systematically introduced
    and informally explained in Sect. 2.

1.1.6.0.6 The Domain Description Language

- The 'Domain Description Language' is `RSL` [39], the `RAISE` `S`pecification `L`anguage [40].
- With suitable, simple adjustments it could also be either of
  - `Alloy` [45],
  - `Event B` [1],
  - `VDM-SL` [30, 31, 37] or
  - `Z` [55].
- We have chosen `RSL` because of its simple provision for
  - defining sorts,
  - expressing axioms, and
  - postulating observers over sorts.

1.1.6.0.7 Domain Descriptions: Narration & Formalisation

- Descriptions
  - *must* be readable and
  - *should* be mathematically precise.[7]
- For that reason we decompose domain description fragments into clearly identified "pairs" of
  - narrative texts and
  - formal texts.

[7] One must insist on formalised domain descriptions in order to be able to verify that domain descriptions satisfy a number of properties not explicitly formulated as well as in order to verify that requirements prescriptions satisfy domain descriptions.

1.1.7. One Domain – Many Models ?

- Will two or more domain engineers cum scientists arrive at “the same domain description” ?
- No, almost certainly not !
- What do we mean by “the same domain description” ?
  - To each proper description we can associate a mathematical meaning, its semantics.
  - Not only is it very unlikely that the syntactic form of the domain descriptions are the same or even “marginally similar”.
  - But it is also very unlikely that the two (or more) semantics are the same;
  - that is, that all properties that can be proved for one domain model can be proved also for the other, and vice versa.

- Why will different domain models emerge ?
  - Two different domain describers will, undoubtedly,
  - when analysing and describing independently,
  - focus on different aspects of the domain.
    - One describer may focus attention on certain phenomena,
    - different from those chosen by another describer.
    - One describer may choose some abstractions
    - where another may choose more concrete presentations.
    - Etcetera.

- We can thus expect that a set of domain description developments lead to a set of distinct models.
  - As these domain descriptions
    - are communicated amongst domain engineers cum scientists
    - we can expect that iterated domain description developments
    - within this group of developers
    - will lead to fewer and more similar models.
  - Just like physicists,
    - over the centuries of research,
    - have arrived at a few models of nature,
    - we can expect there to develop some consensus model of "standard" domains.

- We expect, that sometime in future, software engineers,
  - when commencing software development for a “standard domain”, that is,
  - one for which there exists one or more “standard models”,
  - will start with the development of a domain description
  - based on “one of the standard models” —
  - just like control engineers of automatic control
  - “repeat” an essence of a domain model for a control problem.

1.1.8. Formal Concept Analysis

- Domain analysis involves that of concept analysis.
- As soon as we have identified an entity for analysis we have identified a concept.
  - The entity is a spatio-temporal, i.e., a physical thing.
  - Once we speak of it, it becomes a concept.
- Instead of examining just one entity the domain analyser shall examine many entities.
- Instead of describing one entity the domain describer shall describe a class of entities.
- Ganter & Wille's [38] addresses this issue.

1.1.8.1. A Formalisation

Some Notation:

- By $\mathcal{E}$ we shall understand the type of entities;
- by $\mathbb{E}$ we shall understand an entity of type $\mathcal{E}$;
- by $\mathcal{Q}$ we shall understand the type of qualities;
- by $\mathbb{Q}$ we shall understand a quality of type $\mathcal{Q}$;
- by $\mathcal{E}$**-set** we shall understand the type of sets of entities;
- by $\mathbb{ES}$ we shall understand a set of entities of type $\mathcal{E}$**-set**;
- by $\mathcal{Q}$**-set** we shall understand the type of sets of qualities; and
- by $\mathbb{QS}$ we shall understand a a set of qualities of type $\mathcal{Q}$**-set**.

Definition: 1 Formal Context:

- A **formal context** $\mathbb{K} := (\mathbb{ES}, \mathbb{I}, \mathbb{QS})$ consists of two sets;
  - $\mathbb{ES}$ of entities and
  - $\mathbb{QS}$ of qualities,

  and a
  - relation $\mathbb{I}$ between $\mathbb{E}$ and $\mathbb{Q}$. ■

- To express that $\mathbb{E}$ is in relation $\mathbb{I}$ to a Quality $\mathbb{Q}$ we write
  - $\mathbb{E} \cdot \mathbb{I} \cdot \mathbb{Q}$, which we read as
  - *"entity $\mathbb{E}$ **has** quality $\mathbb{Q}$".*

- Example endurant entities are
  - a specific vehicle,
  - another specific vehicle,
  - etcetera;
  - a specific street segment (link),
  - another street segment,
  - etcetera;
  - a specific road intersection (hub),
  - another specific road intersection,
  - etcetera,
  - a monitor.
- Example endurant entity qualities are
  - (a vehicle) has mobility,
  - (a vehicle) has velocity ($\geq$0),
  - (a vehicle) has acceleration,
  - etcetera;
  - (a link) has length (>0),
  - (a link)has location,
  - (a link)has traffic state,
  - etcetera.

Definition: 2 Qualities Common to a Set of Entities:

- For any subset, $s\mathbb{ES} \subseteq \mathbb{ES}$, of entities we can define $\mathcal{DQ}$ for "derive[d] set of qualities".

$$\mathcal{DQ} : \mathcal{E}\textbf{-set} \rightarrow (\mathcal{E}\textbf{-set} \times \mathcal{I} \times \ \mathcal{Q}\textbf{-set}) \rightarrow \mathcal{Q}\textbf{-set}$$
$$\mathcal{DQ}(s\mathbb{ES})(\mathbb{ES},\mathbb{I},\mathbb{QS}) \equiv \{\mathbb{Q} \mid \mathbb{Q}{:}\mathcal{Q}, \mathbb{E}{:}\mathcal{E} \cdot \mathbb{E}{\in}s\mathbb{ES} \wedge \mathbb{E} \cdot \mathbb{I} \cdot \mathbb{Q}\}$$
$$\textbf{pre}: \ s\mathbb{ES} \subseteq \mathbb{ES}$$

The above expresses: *"the set of qualities common to entities in $s\mathbb{ES}$".* ■

Definition: 3 Entities Common to a Set of Qualities:

- For any subset, $s\mathbb{QS} \subseteq \mathbb{QS}$, of qualities we can define $\mathcal{DE}$ for "derive[d] set of entities".

$$\mathcal{DE}\text{: } \mathcal{Q}\textbf{-set} \to (\mathcal{E}\textbf{-set} \times \mathcal{I} \times \mathcal{Q}\textbf{-set}) \to \mathcal{E}\textbf{-set}$$
$$\mathcal{DE}(s\mathbb{QS})(\mathbb{ES},\mathbb{I},\mathbb{QS}) \equiv \{\mathbb{E} \mid \mathbb{E}{:}\mathcal{E},\ \mathbb{Q}{:}\mathcal{Q} \cdot \mathbb{Q}{\in}s\mathbb{Q} \wedge \mathbb{E} \cdot \mathbb{I} \cdot \mathbb{Q}\},$$
$$\textbf{pre}\text{: } s\mathbb{QS} \subseteq \mathbb{QS}$$

The above expresses: *"the set of entities which have all qualities in $s\mathbb{QS}$".* ■

Definition: 4 Formal Concept:

- A **formal concept** of a context $\mathbb{K}$ is a pair:
  - $(s\mathbb{Q}, s\mathbb{E})$ where
    - $\mathcal{DQ}(s\mathbb{E})(\mathbb{E}, \mathbb{I}, \mathbb{Q}) = s\mathbb{Q}$ and
    - $\mathcal{DE}(s\mathbb{Q})(\mathbb{E}, \mathbb{I}, \mathbb{Q}) = s\mathbb{E}$;
  - $s\mathbb{Q}$ is called the **intent** of $\mathbb{K}$ and $s\mathbb{E}$ is called the **extent** of $\mathbb{K}$. ■

1.1.8.2. Types Are Formal Concepts

- Now comes the "crunch":
  - *In the `TripTych` domain analysis*
  - *we strive to find formal concepts*
  - *and, when we think we have found one,*
  - *we assign a type (or a sort)*
  - *and qualities to it !*

1.1.8.3. Practicalities

- There is a little problem.
  - To search for all those entities of a domain
  - which each have the same sets of qualities
  - is not feasible.
- So we do a combination of two things:
  - we identify a small set of entities
    - all having the same qualities
    - and tentatively associate them with a type, and
  - we identify certain nouns of our national language
    - and if such a noun
      - does indeed designate a set of entities
      - all having the same set of qualities
    - then we tentatively associate the noun with a type.

- Having thus, tentatively, identified a type
  - we conjecture that type
  - and search for counterexamples,
    - that is, entities which
    - refutes the conjecture.
- This "process" of conjectures and refutations is iterated
  - until some satisfaction is arrived at
  - that the postulated type constitutes a reasonable conjecture.

1.1.8.4. Formal Concepts: A Wider Implication

- The formal concepts of a domain form Galois Connections [38].
  - We gladly admit that this fact is one of the reasons why we emphasise **formal concept analysis**.
  - At the same time we must admit that this seminar does not do justice to this fact.
  - We have experimented with the analysis & description of a number of domains
  - and have noticed such Galois connections
  - but it is, for us, too early to report on this.
- Thus we invite the student to study this aspect of domain analysis.

Dines Bjørner's MAP-i Lecture # 1

End of MAP-i Lecture # 1:
An Overview Of Domain Description

Monday, 25 May 2015: 10:30–11:15