Dines Bjørner's MAP-i Lecture # 2

Parts

Monday, 25 May 2015: 11:30–12:15

1.2. Endurant Entities

- In the rest of this seminar we shall consider entities in the context of their being manifest (i.e., spatio-temporal).

1.2.1. General

Definition 1 . 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.*
  - *We further demand that an entity can be objectively described* ■ [8]

[8] **Definitions** and **examples** are delimited by ■ respectively ■

Analysis Prompt 1 . *is_entity:*

- *The domain analyser analyses "things" (θ) into either entities or non-entities.*
- *The method can thus be said to provide the* **domain analysis prompt***:*
  - ***is_entity*** *— where **is_entity(θ)** holds if θ is an entity* [9]

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

[9] **Analysis** prompt definitions and **description** prompt definitions and schemes are delimited by respectively .

Whither Entities:

- The "demands" that entities
  - be observable and objectively describable

  raises some philosophical questions.
- Are sentiments, like feelings, emotions or "hunches" observable ?
- This author thinks not.
- And, if so, can they be other than artistically described ?
- It seems that
  - psychologically and
  - aesthetically

  "phenomena" appears to lie beyond objective description.
- We shall leave these speculations for later.

1.2.2. Endurants and Perdurants

Definition 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.*

  Were we to "freeze" time
  - *we would still be able to observe the entire endurant* ■

- That is, endurants "reside" in space.
- Endurants are, in the words of Whitehead (1920), continuants.

Example 10 . Traffic System Endurants:

Examples of traffic system endurants are:

- traffic system,
- road nets,
- fleets of vehicles,
- sets of hubs,
- sets of links,
- hubs,
- links and
- vehicles ■

Definition 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, that is,*
  - *where we to freeze time we would only see or touch a fragment of the perdurant* ■
- That is, perdurants "reside" in space and time.
- Perdurants are, in the words of Whitehead(1920), occurrents.

Example 11 . Traffic System Perdurants:
Examples of road net perdurants are:

- insertion and removal of hubs or links (actions),
- disappearance of links (events),
- vehicles entering or leaving the road net (actions),
- vehicles crashing (events) and
- road traffic (behaviour) ■

Analysis Prompt 2 . *is_endurant:*

- *The domain analyser analyses an entity, ϕ, into an endurant as prompted by the* **domain analysis prompt***:*
  - ◈ ***is_endurant*** — *ϕ is an endurant if* ***is_endurant(ϕ)*** *holds.*
- ***is_entity*** *is a* **prerequisite prompt** *for* ***is_endurant*** ■

Analysis Prompt 3 . *is_perdurant:*

- *The domain analyser analyses an entity ϕ into perdurants as prompted by the* **domain analysis prompt***:*
  - ◈ ***is_perdurant*** — *ϕ is a perdurant if* ***is_perdurant(ϕ)*** *holds.*
- ***is_entity*** *is a* **prerequisite prompt** *for* ***is_perdurant*** ■

- In the words of Whitehead(1920) — as communicated by Sowa(2000) —
  - an endurant has stable qualities that enable its various appearances at different times to be recognised as the same individual;
  - a perdurant is in a state of flux that prevents it from being recognised by a stable set of qualities.

Necessity and Possibility:

- It is indeed possible to make the endurant/perdurant distinction.
- But is it necessary ?
- We shall argue that it is 'by necessity' that we make this distinction.
  - Space and time are fundamental notions.
  - They cannot be dispensed with.
  - So, to describe manifest domains without resort to space and time is not reasonable.

1.2.3. Discrete and Continuous Endurants

Definition 4 . Discrete Endurant:

- *By a* **discrete endurant** *we shall understand an endurant which is*
  - *separate,*
  - *individual or*
  - *distinct*

  in form or concept ■

Example 12 . Discrete Endurants:

- Examples of discrete endurants are
  - a road net,
  - a hub,
  - a traffic signal,
  - a link,
  - a vehicle,
  - etcetera

Definition 5 . Continuous Endurant:

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

Example 13 . Continuous Endurants:

- Examples of continuous endurants are
  - water,
  - gas,
  - grain,
  - oil,
  - sand,
  - etcetera ■

Analysis Prompt 4 . *is_discrete:*

- *The domain analyser analyse endurants e into discrete entities as prompted by the* **domain analysis prompt***:*
  - ◈ ***is_discrete*** — *e is discrete if* ***is_discrete****(e) holds* ■

Analysis Prompt 5 . *is_continuous:*

- *The domain analyser analyse endurants e into continuous entities as prompted by the* **domain analysis prompt***:*
  - ◈ ***is_continuous*** — *e is continuous if* ***is_continuous****(e) holds* ■

1.2.4. Parts, Components and Materials

1.2.4.1. General

Definition 6 . Part:

- *By a* **part** *we shall understand*
  - *a discrete endurant*
  - *which the domain engineer chooses*
  - *to endow with* **internal qualities** *such as*
    - *unique identification,*
    - *mereology, and*
    - *one or more attributes* ■

We shall define the terms 'unique identification', 'mereology', and 'attributes' shortly.

Example 14 . Parts: Example

- 10 on Slide 84 illustrated,

and examples

- 18 on Slide 109 and
- 19 on Slide 111 illustrate

parts ■

Definition 7 . Component:

- *By a* **component** *we shall understand*
  - *a discrete endurant*
  - *which we, the domain analyser cum describer chooses*
  - *to* ***not*** *endow with* **internal qualities** ■

Example 15 . Components:

- Examples of components are:
  - chairs, tables, sofas and book cases in a living room,
  - letters, newspapers, and small packages in a mail box,
  - machine assembly units on a conveyor belt,
  - boxes in containers of a container vessel,
  - etcetera ■

"At the Discretion of the Domain Engineer":

- We emphasise the following analysis and description aspects:
  - (a) The domain is full of observable phenomena.
    - It is the decision of the domain analyser cum describer
    - whether to analyse and describe some such phenomena,
    - that is, whether to include them in a domain model.
  - (b) The borderline between an endurant
    - being (considered) discrete or
    - being (considered) continuous
    - is fuzzy.
    - It is the decision of the domain analyser cum describer
    - whether to model an endurant as discrete or continuous.

- (c) The borderline between a discrete endurant
  - being (considered) a part or
  - being (considered) a component
  - is fuzzy.
  - It is the decision of the domain analyser cum describer
  - whether to model a discrete endurant as a part or as a component.
- (d) We shall later show how to "compile" parts into processes.
  - A factor, therefore, in determining whether
  - to model a discrete endurant as a part or as a component
  - is whether we may consider a discrete endurant as also representing a process.

Definition 8 . Material:

- *By a* **material** *we shall understand a continuous endurant*

Example 16 . Materials: Examples of material endurants are:

- air of an air conditioning system,
- grain of a silo,
- gravel of a barge,
- oil (or gas) of a pipeline,
- sewage of a waste disposal system, and
- water of a hydro-electric power plant.

Example 17 . Parts Containing Materials:

- Pipeline units are here considered discrete, i.e., parts.
- Pipeline units serve to convey material ■

1.2.4.2. Part, Component and Material Prompts

Analysis Prompt 6 . *is_part:*

- *The domain analyser analyse endurants e into part entities as prompted by the* **domain analysis prompt***:*
  - **is_part** — *e is a part if* **is_part**(*e*) *holds* ■

- We remind the reader that the outcome of `is_part`(*e*)
- is very much dependent on the domain engineer's intention
- with the domain description, cf. Slide 99.

Analysis Prompt 7 . *is_component:*

- *The domain analyser analyse endurants e into component entities as prompted by the* **domain analysis prompt***:*
  - **is_component** *— e is a component if* **is_component**(*e*) *holds*

■

- We remind the reader that the outcome of `is_component`(*e*)
- is very much dependent on the domain engineer's intention
- with the domain description, cf. Slide 99.

Analysis Prompt 8 . *is_material:*

- *The domain analyser analyse endurants e into material entities as prompted by the* **domain analysis prompt***:*
  - ***is_material*** *— e is a material if* ***is_material(e)*** *holds* ■
- We remind the reader that the outcome of `is_material(e)`
- is very much dependent on the domain engineer's intention
- with the domain description, cf. Slide 99.
-

1.2.5. Atomic and Composite Parts

- A distinguishing quality
  - of parts,
  - is whether they are
    - atomic or
    - composite.
- Please note that we shall,
  - in the following,
  - examine the concept of parts
  - in quite some detail.

- That is,
  - parts become the domain endurants of main interest,
  - whereas components and materials become of secondary interest.
- This is a choice.
  - The choice is based on pragmatics.
  - It is still the domain analyser cum describers' choice
    - whether to consider a discrete endurant
    - a part
    - or a component.
  - If the domain engineer wishes to investigate
    - the details of a discrete endurant
    - then the domain engineer choose to model
    - the discrete endurant as a part
    - otherwise as a component.

Definition 9 . Atomic Part:

- **Atomic part**s *are those which,*
  - *in a given context,*
  - *are deemed to not consist of meaningful, separately observable proper* sub-parts ■
- A **sub-part** is a part ■

Example 18 . Atomic Parts: Examples of atomic parts of the above mentioned domains are:

- aircraft (of air traffic),
- demand/deposit accounts (of banks),
- containers (of container lines),
- documents (of document systems),
- hubs, links and vehicles (of road traffic),
- patients, medical staff and beds (of hospitals),
- pipes, valves and pumps (of pipeline systems), and
- rail units and locomotives (of railway systems) ■

Definition 10 . Composite Part:

- **Composite part**s *are those which,*
  - *in a given context,*
  - *are deemed to indeed consist of meaningful, separately observable proper* sub-parts ■

Example 19 . Composite Parts: Examples of atomic parts of the above mentioned domains are:

- airports and air lanes (of air traffic),
- banks (of a financial service industry),
- container vessels (of container lines),
- dossiers of documents (of document systems),
- routes (of road nets),
- medical wards (of hospitals),
- pipelines (of pipeline systems), and
- trains, rail lines and train stations (of railway systems). ■

Analysis Prompt 9 . *is_atomic:*

- *The domain analyser analyses a discrete endurant, i.e., a part p into an atomic endurant:*
  - ***is_atomic**(p): p is an atomic endurant if **is_atomic**(p) holds* ■

Analysis Prompt 10 . *is_composite:*

- *The domain analyser analyses a discrete endurant, i.e., a part p into a composite endurant:*
  - ***is_composite**(p): p is a composite endurant if **is_composite**(p) holds* ■

- `is_discrete` is a **prerequisite prompt** of both `is_atomic` and `is_composite`.

Whither Atomic or Composite:

- If we are analysing & describing vehicles in the context of a road net, cf. the Traffic System Example Slide 84,
  - then we have chosen to abstract vehicles
  - as atomic;
- if, on the other hand, we are analysing & describing vehicles in the context of an automobile maintenance garage
  - then we might very well choose to abstract vehicles
  - as composite —
  - the sub-parts being the object of diagnosis
  - by the auto mechanics.

1.2.6. On Observing Part Sorts

1.2.6.1. Types and Sorts

- We use the term 'sort'
  - when we wish to speak of an abstract type,
  - that is, a type for which we do not wish to express a model[10].
  - We shall use the term 'type' to cover both
    - abstract types and
    - concrete types.

10

- for example, in terms of the concrete types:
  - sets,
  - lists,
  - Cartesians,
  - maps,

  or other.

1.2.6.2. On Discovering Part Sorts

- Recall from the section on *Types Are Formal Concepts* (Slide 76) that we "equate" a formal concept with a type (i.e., a sort).
  - Thus, to us, a part sort is a set of all those entities
  - which all have exactly the same qualities.
- Our aim now
  - is to present the basic principles that let
  - the domain analyser decide on part sorts.

- We observe parts one-by-one.
  - *(α) Our analysis of parts concludes when we have*
    - *"lifted" our examination of a particular part instance*
    - *to the conclusion that it is of a given sort,*
    - *that is, reflects, or is, a formal concept.*
- Thus there is, in this analysis, a "eureka",
  - a step where we shift focus
  - from the concrete to the abstract,
  - from observing specific part instances
  - to postulating a sort:
    - from one to the many.

Analysis Prompt 11 . *observe_parts:*

- *The* **domain analysis prompt***:*
  - ***observe_parts***(p)
- *directs the domain analyser to observe the sub-parts of* p ■

Let us say the sub-parts of p are: $\{p_1,p_2,\ldots,p_m\}$

- *(β) The analyser analyses, for each of these parts, p_{i_k},*
  - *which formal concept, i.e., sort, it belongs to;*
  - *let us say that it is of sort P_k;*
  - *thus the sub-parts of p are of sorts $\{P_1,P_2,\ldots,P_m\}$.*
- *Some P_k may be atomic sorts, some may be composite sorts.*

- The domain analyser continues to examine a finite number of other composite parts: $\{p_j, p_\ell, \ldots, p_n\}$.
  - It is then "discovered", that is, decided, that they all consists of the same number of sub-parts
    - $\{p_{i_1}, p_{i_2}, \ldots, p_{i_m}\}$,
    - $\{p_{j_1}, p_{j_2}, \ldots, p_{j_m}\}$,
    - $\{p_{\ell_1}, p_{\ell_2}, \ldots, p_{\ell_m}\}$,
    - ...,
    - $\{p_{n_1}, p_{n_2}, \ldots, p_{n_m}\}$,

    of the same, respective, part sorts.
  - *(γ) It is therefore concluded, that is, decided, that $\{p_i, p_j, p_\ell, \ldots, p_n\}$ are all of the same part sort P with observable part sub-sorts $\{P_1, P_2, \ldots, P_m\}$.*

- Above we have ***type-font-highlighted*** three sentences: (α, β, γ).
- When you analyse what they "prescribe" you will see that they entail a "depth-first search" for part sorts.
  - The β sentence says it rather directly:
  - ***"The analyser analyses, for each of these parts,*** p_k***, which formal concept, i.e., part sort it belongs to."***
  - To do this analysis in a proper way, the analyser must ("recursively") analyse the parts "down" to their atomicity,
  - and from the atomic parts decide on their part sort,
  - and work ("recurse") their way "back",
  - through possibly intermediate composite parts,
  - to the p_ks.

1.2.6.3. Part Sort Observer Functions

- The above analysis amounts to the analyser
  - first "applying" the domain analysis prompt
  - is_composite(p) to a discrete endurant,
  - where we now assume that the obtained truth value is **true**.
  - Let us assume that parts p:P consists of sub-parts of sorts $\{P_1, P_2, \ldots, P_m\}$.
  - Since we cannot automatically guarantee that our domain descriptions secure that
    - P and each P_i ([$1 \leq i \leq$m])
    - denotes disjoint sets of entities

    we must prove it.

Domain Description Prompt 1 . *observe_part_sorts*:

- *If* ***is_composite****(p) holds, then the analyser "applies" the description language observer prompt*
  - ***observe_part_sorts****(p)*

  resulting in the analyser writing down the ***part sorts and part sort observers*** *domain description text according to the following schema:*

1. observe_part_sorts schema

Narration:

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

Formalisation:

type
[s] P,
[s] P_i $[1 \leq i \leq m]$ **comment:** P_i $[1 \leq i \leq m]$ abbreviates P_1, P_2, ..., P_m
value
[o] **obs_part**_P_i: P → P_i $[1 \leq i \leq m]$
[i] **is**_P_i: P_i → **Bool** $[1 \leq i \leq m]$

proof obligation [Disjointness of part sorts]
[p] $\forall\ p{:}(P_1|P_2|...|P_m) \cdot$
[p] $\bigwedge$ {**is**_P_i(p) ≡ $\bigvee$ ∼ {**is**_P_j(p) | j ∈ {1..m} \ {i}} | i ∈ {1..m}}

Example 20 . Composite and Atomic Part Sorts of Transportation:

- The following example illustrates the multiple use of the `observe_part_sor` function:
  - first to δ, a specific transport domain, Item 12,
  - then to an $n : N$, the net of that domain, Item 13, and
  - then to an $f : F$, the fleet of that domain, Item 14.

12 A transportation domain is composed from a net, a fleet (of vehicles) and a monitor.

13 A transportation net is composed from a collection of hubs and a collection of links.

14 A fleet is a collection of vehicles.

- The monitor is considered an atomic part.

type
12. N, F, M
value
12. **obs_part**_N:Δ→N, **obs_part**_F:Δ→F, **obs_part**_M:Δ→M
type
13. HC, LC
value
13. **obs_part**_HC:N→HC, **obs_part**_LC:N→LC
type
14. VC
value
14. **obs_part**_VC:F→VC

- *A proof obligation has to be discharged,*
  - *one that shows disjointedness of sorts* N, F *and* M.
  - *An informal sketch is:*
    - *entities of sort* N *are composite and consists of two parts:*
    - *aggregations of hubs,* HS, *and aggregations of links,* LS.
    - *Entities of sort* F *consists of an aggregation,* VS, *of vehicles.*
    - *So already that makes* N *and* F *disjoint.*
    - M *is an atomic entity — where* N *and* F *are both composite.*
    - *Hence the three sorts* N, F *and* M *are disjoint* ■

1.2.6.4. On Discovering Concrete Part Types

Analysis Prompt 12 . *has_concrete_type:*

- *The domain analyser*
  - *may decide that it is expedient, i.e., pragmatically sound,*
  - *to render a part sort, P, whether atomic or composite, as a concrete type, T.*
  - *That decision is prompted by the holding of the* **domain analysis prompt***:*
    - *has_concrete_type(p).*
  - *is_discrete is a* **prerequisite prompt** *of has_concrete_type*

- The reader is reminded that
  - the decision as to whether an abstract type is (also) to be described concretely
  - is entirely at the discretion of the domain engineer.

Domain Description Prompt 2 . *observe_part_type*:

- *Then the domain analyser applies the* **domain description prompt**:
  - ***observe_part_type****(p)*[11]
- *to parts p:P which then yield the* part type and part type observers *domain description text according to the following schema:*

[11] `has_concrete_type` is a **prerequisite prompt** of `observe_part_type`.

2. observe_part_type schema

Narration:

[t_1] … narrative text on sorts and types S_i …

[t_2] … narrative text on types T …

[o] … narrative text on type observers …

Formalisation:

type

[t_1] S_1, S_2, …, S_m, …, S_n,

[t_2] $T = \mathcal{E}(S_1, S_2, \ldots, S_n)$

value

[o] **obs_part**_T: P $\rightarrow$ T

- The type names,
  - T, of the concrete type,
  - as well as those of the auxiliary types, S_1,S_2,...,S_m,
  - are chosen by the domain describer:
    - they may have already been chosen
    - for other sort–to–type descriptions,
    - or they may be new.

Example 21 . Concrete Part Types of Transportation:

We continue Example 20 on Slide 123:

15 A collection of hubs is a set of hubs and a collection of links is a set of links.

16 Hubs and links are, until further analysis, part sorts.

17 A collection of vehicles is a set of vehicles.

18 Vehicles are, until further analysis, part sorts.

type
15. Hs = H-**set**, Ls = L-**set**
16. H, L
17. Vs = V-**set**
18. V

value
15. **obs_part**_Hs:HC→Hs, **obs_part**_Ls:LC→Ls
17. **obs_part**_Vs:VC→Vs ■

1.2.6.5. Forms of Part Types

- Usually it is wise to restrict the part type definitions, $\mathsf{T}_i = \mathcal{E}_i(\mathsf{Q,R,...,S})$, to simple type expressions.

  - $\mathsf{T{=}A\text{-}\mathbf{set}}$ or
  - $\mathsf{T{=}A^*}$ or
  - $\mathsf{T{=}ID} \underset{m}{\to} \mathsf{A}$ or
  - $\mathsf{T{=}A}_t|\mathsf{B}_t|...|\mathsf{C}_t$

  where

  - ID is a sort of unique identifiers,
  - $\mathsf{T{=}A}_t|\mathsf{B}_t|...|\mathsf{C}_t$ defines the disjoint types
    - $\mathsf{A}_t{==}\mathsf{mkA}_s(\mathsf{s{:}A}_s)$,
    - $\mathsf{B}_t{==}\mathsf{mkB}_s(\mathsf{s{:}B}_s)$, ...,
    - $\mathsf{C}_t{==}\mathsf{mkC}_s(\mathsf{s{:}C}_s)$,

    and where
  - A, A_s, B_s, ..., C_s are sorts.
  - Instead of $\mathsf{A}_t{==}\mathsf{mkA}(\mathsf{a{:}A}_s)$, etc., we may write $\mathsf{A}_t{::}\mathsf{A}_s$ etc.

1.2.6.6. Part Sort and Type Derivation Chains

- Let P be a composite sort.
- Let P_1, P_2, ..., P_m be the part sorts "discovered" by means of `observe_part_sorts(p)` where p:P.
- We say that P_1, P_2, ..., P_m are (immediately) **derived** from P.
- If P_k is derived from P_j and P_j is derived from P_i, then, by transitivity, P_k is **derived** from P_i.

1.2.6.6.1 No Recursive Derivations

- We "mandate" that
  - if P_k is derived from P_j
  - then there
    - can be no P derived from P_j
    - such that P is P_j,
    - that is, P_j cannot be derived from P_j.
- That is, we do not allow recursive domain sorts.
- It is not a question, actually of allowing recursive domain sorts.
  - It is, we claim to have observed,
  - in very many domain modeling experiments,
  - that there are no recursive domain sorts!

1.2.6.7. Names of Part Sorts and Types

- The domain analysis and domain description text prompts
  - `observe_part_sorts`,
  - `observe_material_sorts` and
  - `observe_part_type`

  — as well as the
  - `attribute_names`,
  - `observe_material_sorts`,
  - `observe_unique_identifier`,
  - `observe_mereology` and
  - `observe_attributes`

  prompts introduced below — "yield" type names.
  - That is, it is as if there is
    - a reservoir of an indefinite-size set of such names
    - from which these names are "pulled",
    - and once obtained are never "pulled" again.

- There may be domains for which two distinct part sorts may be composed from identical part sorts.
- In this case the domain analyser indicates so by prescribing a part sort already introduced.

Example 22 . Container Line Sorts:

- Our example is that of a container line
  - with container vessels and
  - container terminal ports.

19 A container line contains a number of container vessels and a number of container terminal ports, as well as other components.

20 A container vessel contains a container stowage area, etc.

21 A container terminal port contains a container stowage area, etc.

22 A container stowage ares contains a set of uniquely identified container bays.

23 A container bay contains a set of uniquely identified container rows.

24 A container row contains a set of uniquely identified container stacks.

25 A container stack contains a stack, i.e., a first-in, last-out sequence of containers.

26 Containers are further undefined.

- After a some slight editing we get:

type
CL
VS, VI, V, Vs = VI $\overrightarrow{m}$ V,
PS, PI, P, Ps = PI $\overrightarrow{m}$ P
value
obs_part_VS: CL → VS
obs_part_Vs: VS → Vs
obs_part_PS: CL → PS
obs_part_Ps: CTPS → CTPs
type
CSA
value
obs_part_CSA: V → CSA
obs_part_CSA: P → CSA

type
BAYS, BI, BAY, Bays=BI $\overrightarrow{m}$ BAY
ROWS, RI, ROW, Rows=RI $\overrightarrow{m}$ ROW
STKS, SI, STK, Stks=SI $\overrightarrow{m}$ STK
C
value
obs_part_BAYS: CSA → BAYS,
obs_part_Bays: BAYS → Bays
obs_part_ROWS: BAY → ROWS,
obs_part_Rows: ROWS → Rows
obs_part_STKS: ROW → STKS,
obs_part_Stks: STKS → Stks
obs_part_Stk: STK → C^*

- Note that `observe_part_sorts`(v:V) and `observe_part_sorts`(p:P) both yield CSA ■

1.2.6.8. More On Part Sorts and Types

- The above "experimental example" motivates the below.
  - We can always assume that composite parts $p{:}P$ abstractly consists of a definite number of sub-parts.
    - **Example 23**. We comment on Example 20 on Slide 123: parts of type Δ and N are composed from three, respectively two abstract sub-parts of distinct types ■
  - Some of the parts, say p_{i_z} of $\{p_{i_1},p_{i_2},\ldots,p_{i_m}\}$, of $p{:}P$, may themselves be composite.
    - **Example 24**. We comment on Example 20 on Slide 123: parts of type N, F, HC, LC and VC are all composite ■

- There are, pragmatically speaking, two cases for such compositionality.
  - Either the part, p_{i_z}, of type t_{i_z}, is is composed from a definite number of abstract or concrete sub-parts of distinct types.
    - **Example 25.** We comment on Example 20 on Slide 123: parts of type N are composed from three sub-parts ■
  - Or it is composed from an indefinite number of sub-parts of the same sort.
    - **Example 26.** We comment on Example 20 on Slide 123: parts of type HC, LC and VC are composed from an indefinite numbers of hubs, links and vehicles, respectively ■

Example 27 . Pipeline Parts:

27 A pipeline consists of an indefinite number of pipeline units.

28 A pipeline units is either a well, or a pipe, or a pump, or a valve, or a fork, or a join, or a sink.

29 All these unit sorts are atomic and disjoint.

type
27. PL, U, We, Pi, Pu, Va, Fo, Jo, Si
27. Well, Pipe, Pump, Valv, Fork, Join, Sink
value
27. **obs_part**_Us: PL → U-**set**
type
28. U == We | Pi | Pu | Va | Fo | Jo | Si
29. We::Well, Pi::Pipe, Pu::Pump, Va::Valv, Fo:Fork, Jo::Join, Si::Sink ■

1.2.6.8.1 Derivation Lattices

- Derivation chains
  - start with the domain name, say Δ, and
  - (definitively) end with the name of an atomic sort.
- Sets of derivation chains form **join lattice**s [3].

Example 28 . Derivation Chains:

- Figure 1 on the following slide illustrates
  - two part sort and type derivation chains.
  - based on Examples 20 on Slide 123 and 22 on Slide 135, respectively.

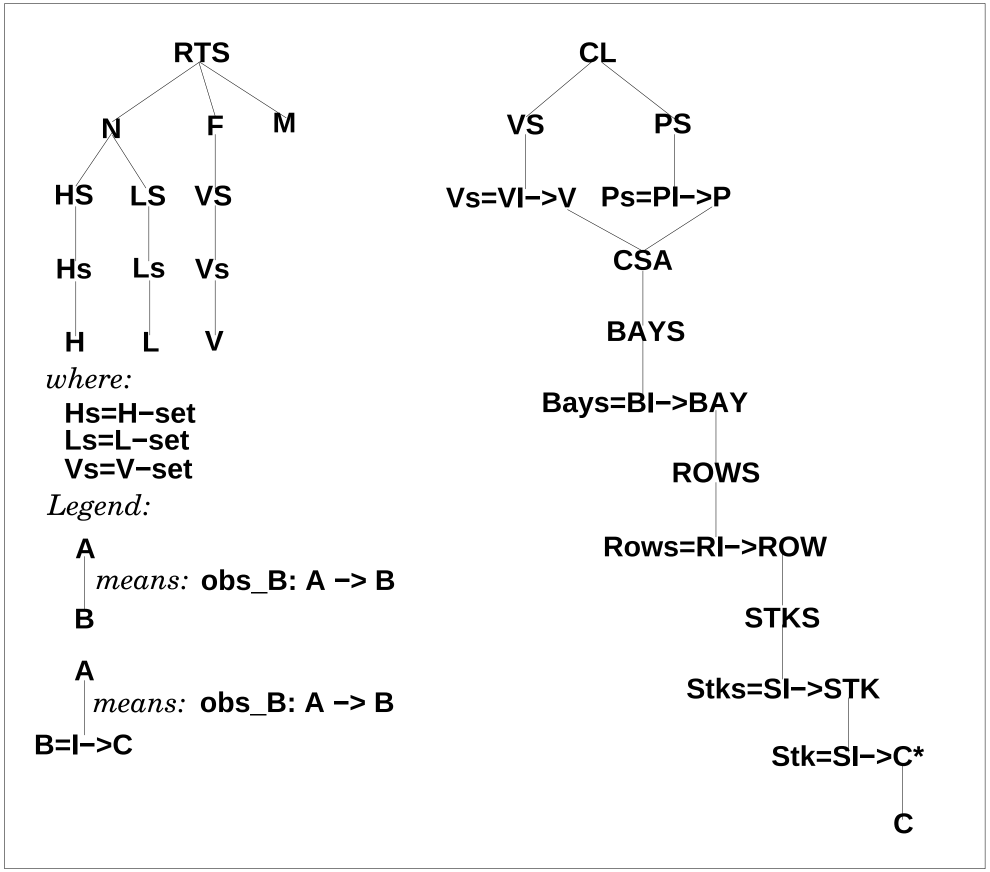

Figure 1: Two Domain Lattices: Examples 20 on Slide 123 and 22 on Slide 135

- The "–>" of Fig. 1 stands for $\overrightarrow{m}$ ■

1.2.6.9. External and Internal Qualities of Parts

- By an **external part quality** we shall understand the
  - `is_atomic`,
  - `is_composite`,
  - `is_discrete` and
  - `is_continuous`

  qualities.
- By an **internal part quality** we shall understand the part qualities to be outlined in the next sections:
  - `unique ids`,
  - `mereology` and
  - `attributes`.
- By **part qualities** we mean the sum total of
  - `external endurant` and
  - `internal endurant`

  qualities.

1.2.6.10. Three Categories of Internal Qualities

- We suggest that the internal qualities of parts be analysed into three categories:
  - (i) a category of unique part identifiers,
  - (ii) a category of mereological quantities and
  - (iii) a category of general attributes.

- Part mereologies are about **sharing** qualities between parts.
  - Some such **sharing** expresses spatio-topological properties of how parts are organised.
  - Other part **sharing** aspects express relations (like equality) of part attributes.
  - We base our modeling of mereologies on the notion of unique part identifiers.
  - Hence we cover **internal qualities** in the order (i–ii–iii).

End of MAP-i Lecture # 2:
Parts

Monday, 25 May 2015: 11:30–12:15