

## Start of Lecture 3: DOMAINS: Intrinsic – Rules & Regulations

Usually a domain description is a set of documents with many parts recording many facets of the domain: The

- *business processes*,
- *intrinsic*,
- *support technology*,
- *rules and regulations*,
- *management and organisation*, and the
- *human behaviours*.

## 3. An Ontology of Domain Facets

### 3.0.1. Definitions

**Definition: Domain.** *An area of activity which some software is to support (or supports) or partially or fully automate (resp. automates).*

- The term ‘application domain’ is considered synonymous with the term ‘domain’.

**Definition: Domain Description.** *A textual, informal or formal document which describes a domain as it is.*

**Definition: Domain Engineering.**

- *The engineering of the development of a domain description, from*
  - *identification of domain stakeholders, via*
  - *domain acquisition,*
  - *domain analysis,*
  - *terminologisation,**and*
  - *domain description**to*
  - *domain validation and*
  - *domain verification.*

## Definition: Domain Facet.

- By a domain facet we understand
  - one amongst a finite set of generic ways of analysing a domain:
  - A view of the domain, such that the different facets cover conceptually different views,
  - and such that these views together cover the domain.
- We consider here the following domain facets:
  - *business processes*,
  - *intrinsic*s,
  - *support technology*,
  - *rules and regulations*,
  - *management and organisation*,
  - and
  - *human behaviour*.

### 3.0.3. Business Processes

#### 3.0.3.1. A Characterisation

- By a business process we shall understand
  - a *behaviour*
  - of an enterprise, a business, an institution, a factory.

#### 3.0.3.2. An Example

- The business processes of transportation evolves around
  - freights or passengers
  - being transported along routes
  - by a vehicle (car, train, aircraft, ship)
  - “propelled” by some locomotive force.

## 3.0.2. What Can Be Observed

- “Whether you can observe a thing or not depends on the theory which you use. It is the theory which decides what can be observed.”
- Albert Einstein objecting to the placing of observables at the heart of the new quantum mechanics, during Heisenberg’s 1926 lecture at Berlin; related by Heisenberg, quoted in *Unification of Fundamental Forces* (1990) by Abdus Salam ISBN 0521371406.

## 3.1. Intrinsic

### Definition: Intrinsic.

- By the *intrinsic*s of a domain we shall understand
  - those phenomena and concepts of a domain
  - which are basic to any of the other facets,
  - with such a domain intrinsic initially covering at least one stakeholder view.

### 3.1.1. Net Topology Descriptors

Instead of dealing with the entire phenomenon of a net, that is, the real, physical, geographic “thing”, we can describe essentials of a net, for example how its hub and links are connected.

56. One way of abstractly modelling a net descriptor is as a map, **nd**, from hub identifiers to simple maps, **lihis**, from link identifiers to hub identifiers,
57. such that
- for all **hi** in (the definition set of) **nd** it is the case that
  - if **hi** maps to **lihi**,
  - and in that link identifier to hub identifier map, **li** maps to **hi'**,
  - then **hi'** is different from **hi** and
  - hi'** maps to an **lihi'** in which **li** is defined and maps to **hi**.
  - And there are only such pairings.

From a net one can construct its net descriptor:

**value**

```
conND: N → ND
conND(n) ≡
  [ hi → [ li → hi' | li:LI, hi':HI, li ∈ obs_LLs(getH(hi,n)) ∧ {hi, hi'} = obs_HIs(getL(li,n)) ] ]
  hi:HI, hi ∈ xtrHIs(n) ]
```

**type**

```
56. ND' = HI  $\xrightarrow{m}$  (LI  $\xrightarrow{m}$  HI)
56. ND = { |nd':ND.wf_ND(nd')| }
```

**value**

```
57. wf_ND: ND' → Bool
57. wf_ND(nd) ≡
57(a).   ∀ hi:HI, hi ∈ dom nd ⇒
57(b).   let lihi = nd(hi) in
57(c).   ∀ li:LI, li ∈ dom lihi ⇒
57(c).   let hi' = (nd(hi))(li) in
57(d).   hi ≠ hi' ∧
57(e).   hi' ∈ dom nd ∧ li ∈ dom(nd(hi')) ∧ hi = (nd(hi'))(li)
57(f).   end end
```

### 3.1.2. Link States and Link State Spaces

- We introduce the notions of
  - the state of a link,
  - the state of a hub,
  - the state space of a link and
  - the state space of a hub.
- States abstract directions of movement.
- Links are, by our previous definitions, bi-directional:
  - from one of the connected hubs to the other,
  - and vice versa.
- And hubs are multi-directional:
  - from potentially any link via the hub to potentially any link.

- Let
  - the observed hub identifiers of a link  $\ell$  be  $\{h_j, h_k\}$ ,
  - then link  $\ell$  can potentially be in any one of the four link states:
    - $\{(h_j, h_k), (h_k, h_j)\}$ ,  $\{(h_j, h_k)\}$ ,  $\{(h_k, h_j)\}$  and  $\{\{\}\}$ .
- Any one particular link may
  - always remain in one and the same state,
  - or it may from time to time undergo transitions between any subset of the potential link state space.

## type

58.  $L\Sigma = (\text{HI} \times \text{HI})\text{-set}$

59.  $L\Omega = L\Sigma\text{-set}$

## value

60. `generate_full_LSigma`:  $L \rightarrow L\Sigma$

60. `generate_full_LSigma(l)`  $\equiv$

60.  $\{\{\} \cup \{(hi', hi'') \mid hi', hi'' : \text{HI} \cdot hi' \neq hi'' \wedge \{hi', hi''\} = \text{obs\_HIs}(l)\}\}$

60. `generate_LOmega`:  $L \rightarrow L\Omega$

60. **let** `fullLsigma` = `generate_full_LSigma(l)` **in**

60.  $\{\{\}, \cup \{\sigma \mid \sigma : L\Sigma \cdot \sigma \subseteq \text{fullLsigma}\}\}$  **end**

61. `obs_LSigma`:  $L \rightarrow L\Sigma$

62. `obs_LOmega`:  $L \rightarrow L\Omega\text{-set}$

58. Link states,  $l\sigma : L\Sigma$ , are set of pairs of hub identifiers.

59. Link state spaces are set of link states.

60. From a link one can generate the link state space of all potential link states.

61. From a link one can observe the current link state  $l\sigma : L\Sigma$ .

62. From a link one can observe the link state space  $l\omega : L\Omega$ .

## 3.1.3. Hub States and Hub State Spaces

63. Hub states,  $h\sigma : H\Sigma$ , are sets of pairs of link identifiers  $((l_i, l_k))$ , designating that if  $(l_i, l_k)$  is in the current hub state then movement can take place from the link designated by  $l_i$  (via hub  $h$ ) to the link designated by  $l_k$ .

64. Hub state spaces are set of hub states.

65. From a hub one can generate the hub state space of all potential hub states.

66. From a hub one can observe the current hub state  $h\sigma : H\Sigma$ .

67. From a hub one can observe the hub state space  $h\omega : H\Omega$ .

**type**63.  $H\Sigma = (LI \times LI)\text{-set}$ 64.  $H\Omega = H\Sigma\text{-set}$ **value**65.  $\text{generate\_full\_H}\Sigma: H \rightarrow H\Sigma$ 65.  $\text{generate\_full\_H}\Sigma(h) \equiv$ 65.  $\{\} \cup \{(li', li'') \mid li', li'' : LI \cdot \{li', li''\} \subseteq \text{obs\_LIs}(h)\}$ 60.  $\text{generate\_H}\Omega: H \rightarrow H\Omega$ 60. **let**  $\text{fullH}\sigma = \text{generate\_full\_H}\Sigma(h)$  **in**60.  $\{\{\} \cup \{\sigma \mid \sigma : H\Sigma \cdot \sigma \subseteq \text{fullH}\sigma\}\}$  **end**66.  $\text{obs\_H}\Sigma: H \rightarrow H\Sigma$ 66.  $\text{obs\_H}\Omega: H \rightarrow H\Sigma\text{-set}$ **3.1.5. Concrete Types for Simple Entities**

- As an alternative for, or as a step of refinement from the earlier sorts of nets, hubs and links
- one can simplify matters by concrete types for these simple entities.

70. Nets are Cartesians of sets of hubs and links.

71. A link is a Cartesian of a link identifier, a set of exactly two hub identifiers, a link state, a link state space, and a number of presently further unspecified link attributes.

72. A hub is a Cartesian of a hub identifier, a set of zero, one or more link identifiers, a hub state, a hub state space, and a number of presently further unspecified hub attributes.

**3.1.4. State and State Space Wellformedness**

68. States must be in appropriate state spaces.

69. State spaces must be subsets of all potential appropriate states.

**axiom** $\forall n:N, l:L, h:H \cdot l \in \text{obs\_Ls}(n) \wedge h \in \text{obs\_Hs}(n) \Rightarrow$ 58.  $\text{obs\_L}\Sigma(l) \in \text{obs\_L}\Omega(l) \wedge$ 59.  $\text{obs\_L}\Omega(l) \subseteq \text{generate\_full\_L}\Sigma(l) \wedge$ 58.  $\text{obs\_H}\Sigma(h) \in \text{obs\_H}\Omega(h) \wedge$ 59.  $\text{obs\_H}\Omega(h) \subseteq \text{generate\_full\_H}\Sigma(h)$ **theorems:** $\forall n:N, l:L, h:H \cdot l \in \text{obs\_Ls}(n) \wedge h \in \text{obs\_Hs}(n) \Rightarrow$  $\text{obs\_L}\Sigma(l) \subseteq \{(hi', hi'') \mid hi', hi'' : H \cdot \{hi', hi''\} \subseteq \text{obs\_HIs}(l)\} \wedge$  $\text{obs\_H}\Sigma(h) \subseteq \{(li', li'') \mid li', li'' : L \cdot \{li', li''\} \subseteq \text{obs\_LIs}(h)\}$ **type**70.  $N = H\text{-set} \times L\text{-set}$ 71.  $L :: \text{obs\_LI}:LI \times \text{obs\_HIs}:HI\text{-set} \times L\Sigma \times L\Omega \times LAtr$ 72.  $H :: \text{obs\_HI}:HI \times \text{obs\_LIs}:LI\text{-set} \times H\Sigma \times H\Omega \times HAtr$

We leave it to the reader to narrate the wellformedness constraints.

### axiom

$$\begin{aligned}
 & \forall (hs, ls): N \cdot ls \neq \{\} \Rightarrow \mathbf{card} \ hs \geq 2 \wedge \\
 & \forall l', l'': L \cdot \{l', l''\} \subseteq ls \wedge l' \neq l'' \Rightarrow \mathit{obs\_LI}(l') \neq \mathit{obs\_LI}(l'') \wedge \\
 & \forall h', h'': H \cdot \{h', h''\} \subseteq hs \wedge h' \neq h'' \Rightarrow \mathit{obs\_HI}(h') \neq \mathit{obs\_HI}(h'') \wedge \\
 & \forall l: (li, his, l\sigma, l\omega, latrs): L \cdot l \in ls \Rightarrow \\
 & \quad \mathbf{card} \ his = 2 \wedge his \subseteq \{\mathit{obs\_HI}(h'') \mid h'': H \cdot h'' \in hs\} \wedge \\
 & \quad l\sigma \in \mathit{generate\_full\_L\Sigma}(l) \wedge \\
 & \quad l\omega \subseteq \mathit{generate\_full\_L\Sigma}(l) \wedge \\
 & \forall h: (hi, lis, h\sigma, h\omega, hats): H \cdot h \in hs \Rightarrow \\
 & \quad lis \subseteq \{\mathit{obs\_LI}(l'') \mid l'': L \cdot l'' \in ls\} \wedge \\
 & \quad h\sigma \in \mathit{generate\_full\_H\Sigma}(h) \wedge \\
 & \quad h\omega \subseteq \mathit{generate\_full\_H\Sigma}(h)
 \end{aligned}$$

The top left hub/link diagram (1.) thus can be claimed to depict hub state  $\{(A, B), (A, C), (A, D), (B, C), (C, D), (D, A)\}$ .

Photo 2 shows a semaphore which seems to be able to display all kinds of states.



Figure 2: A General Purpose Traffic Light

The point of this example is to show that a hub may take on many states, that not all hub states may be desirable (viz., lead to crossing traffic if so interpreted), and that to reach from one hub state to another one must change the state.

### 3.1.6. Example Hub Crossings

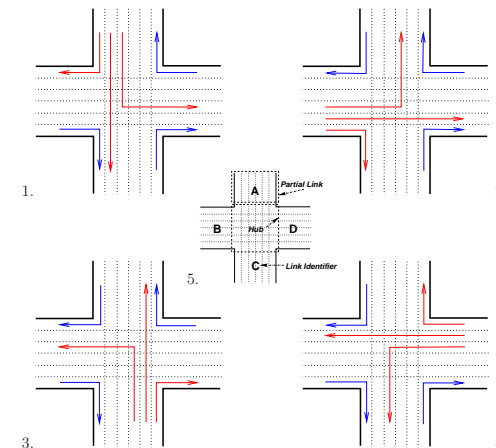


Figure 1: Four "Safe" Flows

### 3.1.7. Actions Continued

73. The action  $\mathit{change\_H\Sigma}$  takes a hub,  $h$ , in some state, and a desired next state,  $h\sigma'$ , and results in a hub,  $h'$ , which

- (a) has the same hub identifier as  $h$ ,
  - is connected to the same links as  $h$ ,
  - has the same hub state space as  $h$ ,
  - has the same attributes (names and values) as  $h$ ,
- (b) but whose state may have changed.

73(b). The new state of  $h'$  ought be  $h\sigma'$ , but electro-mechanical or other failures in setting the state may set the new state to any state of the potential states of  $h$  (i.e.,  $h'$ ), not just to any state in the hub state space of  $h$ .

**value**

73.  $\text{change\_H}\Sigma: H \times H\Sigma \rightarrow H$

73.  $\text{change\_H}\Sigma((hi, lis, h\sigma, h\omega, hats), h\sigma') \equiv$

73(b). **let**  $h\sigma''' \in \text{generate\_full\_H}\Sigma\text{s}$  **in**

73(a).  $(hi, lis, h\sigma''', h\omega, hats)$  **end**

- Had we specified that the resulting state must be  $h\sigma'$
- then we had prescribed a requirements to a **change** operation.
- As it is now we have described a domain phenomenon, namely that operations may fail.

**3.2.1. Traffic Signals**

A traffic signal represents a technology in support of visualising hub states and in effecting state changes.

74. A hub state is now modelled as a triple: the link identifier  $l_i$  (“coming from”), a colour (**red**, **yellow**, and **green**), and another the link identifier  $l_j$  (“going to”).

75. Signalling is now a sequence of one or more pairs of next hub states and time intervals:

$$\langle (h\sigma_1, ti_1), (h\sigma_2, ti_2), \dots, (h\sigma_{n-1}, ti_{n-1}), (h\sigma_n, ti_n) \rangle, n > 0$$

**3.2. Support Technologies**

**Definition: Support Technology.** *By a support technology we understand*

- a facet of a domain,
  - one which reflects its (current) dependency on
    - human,
    - mechanical,
    - electro-mechanical,
    - electronic and/or
    - other technologies
- (i.e., tools) in order to carry out its business processes.

- The idea of a signalling is
  - to first change the designated hub to state  $h\sigma_1$ ,
  - then wait  $ti_1$  time units,
  - then set the designated hub to state  $h\sigma_2$ ,
  - then wait  $ti_2$  time units,
  - etcetera, ending with final state  $\sigma_n$
  - and a (supposedly) long time interval  $ti_n$
  - before any decisions are to be made as to another signalling.
- The set of hub states  $\{h\sigma_1, h\sigma_2, \dots, h\sigma_{n-1}\}$  of
  - $\langle (h\sigma_1, ti_1), (h\sigma_2, ti_2), \dots, (h\sigma_{n-1}, ti_{n-1}), (h\sigma_n, ti_n) \rangle, n > 0$
 are called intermediate states.
- Their purpose is to secure an orderly vehicle-wise safe signal transitions from **red** to **green** etc.

76. A street signal (a semaphore) is now abstracted as a map from pairs of hub states to signalling sequences.

The idea is that given a hub one can observe its semaphore, and given the state,  $h\sigma$  (not in the above set), of the hub “to be signalled” and the state  $h\sigma_n$  into which that hub is to be signalled “one looks up” under that pair in the semaphore and obtains the desired signalling.

### type

74.  $H\Sigma = LI \times Colour \times LI$

74.  $Colour == red \mid yellow \mid green$

75.  $Signalling = (H\Sigma \times TI)^*$

75.  $TI$

76.  $Semaphore = (H\Sigma \times H\Sigma) \xrightarrow{m} Signalling$

### value

76.  $obs\_Semaphore: H \rightarrow Semaphore$

### 3.2.2. Traffic “Control”

78. Given two hub states,  $h\sigma_{init}$  and  $h\sigma_{end}$ , where  $h\sigma_{init}$  designates a present hub state and  $h\sigma_{end}$  designates a desired next hub state after signalling.

79. Now **signalling** is a sequence of one or more successful hub state changes.

### value

78.  $signalling: H\Sigma \times H\Sigma \rightarrow H \rightarrow H$

79.  $signalling(h\sigma_{init}, h\sigma_{end})(h) \equiv$

79. **let** sema = obs\_Semaphore(h) **in**

79. **let** sg = sema(h $\sigma_{init}$ , h $\sigma_{end}$ ) **in**

79. signal\_sequence(sg)(h) **end end**

79. **pre** (h $\sigma_{init}$ , h $\sigma_{end}$ )  $\in \mathbf{dom}$  obs\_Semaphore(h)

77. A hub semaphore, **sema**, contains only such hub states as are observed in the **hub state space**.

(a) Let **hsps** be the set of “from/to” hub state pairs in **sema**.

(b) Then **hs** is the set of all hub states mentioned in **hsps**.

(c) To **hs** join all the hub states mentioned in any signalling, **sg**, of **sema**.

77. hub\_state\_space: Semaphore  $\rightarrow H\Sigma$ -set

77. hub\_state\_space(sema)  $\equiv$

77(a). **let** hsps = {hsp | hsp: (HΣ × HΣ) · hsp  $\in \mathbf{dom}$  sema} **in**

77(b). **let** hs = {hσ', hσ'' | hσ', hσ'': HΣ · (hσ', hσ'')  $\in$  hsps} **in**

77(c). hs  $\cup \cup$  { {hσ | (hσ, ti): (HΣ × TI) · (hσ, ti)  $\in \mathbf{elems}$  sg} | sg: Signalling · sg  $\in$  sema }

77. **end end**

### axiom

77.  $\forall h: H \cdot \cup obs\_H\Omega(h) = hub\_state\_space(obs\_Semaphore(h))$

79. signal\_sequence( $\langle \rangle$ )(h)  $\equiv h$

79. signal\_sequence( $\langle (h\sigma, ti) \rangle^{\wedge} sg$ )(h)  $\equiv$

79. **let** hσ' = change\_HΣ(h)(hσ) **in**

79. **if** hσ'  $\neq$  hσ **then chaos**

79. **else wait**(ti); signal\_sequence(sg)(h) **end end**

- If a desired hub state change fails (**chaos**) then we do not define the outcome of signalling.



### 3.3. Rules and Regulations

**Definition: Rule.** A rule stipulates a regulating principle.

- In the context of modelling domain rules we shall understand a domain rule
  - as some *text*
  - whose meaning is a *predicate*
  - over a pair of suitably chosen domain *states*.
- We may assume that
  - a domain *action* or a domain *event*
  - takes place in the *first* of these *states* and
  - results in the *second* of these *states*.
- If the *predicate* is true
  - then we say that the rule has been *obeyed*,

Usually a domain rule is paired with a possibly remedying regulation.

**Definition: Regulation.**

- A *regulation* stipulates that
  - an *action* be taken
  - in order to remedy a previous action which violated a *rule*.
- That is,
  - a *regulation* is some *text*
  - which designates a possibly composite *action*,
  - that is, a *state-to-state change*
  - which ostensibly results in a *state*
  - in which the *rule*, “attached” to the regulation, *now holds*.

– otherwise that it has been violated.

#### 3.3.1. Vehicles

80. Vehicles are further undefined quantities except that

- (a) vehicles have unique identifiers,
- (b) vehicles are either positioned
  - i. at/in hubs
  - ii. or on links, in some fractional (non-zero) distance from a hub toward the connecting hub.

81. From a net (sort) one can observe all the vehicles of the net.<sup>1</sup>

82. No two vehicles so observed have the same identifier.

<sup>1</sup>Thus a concrete net type, in addition to hubs and links (now) also contains vehicles.

**type**

80. V  
 80(a). VI  
 80(b). VP = HP | LP  
 80((b)i). HP == atH(hi:HI)  
 80((b)ii). LP == onL(li:LI,fhi:HI,f:F,thi:HI)  
 80((b)ii). F = {f:F·0<f<1}

**value**

- 80(a). obs\_VI: V → VI  
 80(b). obs\_VP: V → VP  
 81. obs\_Vs: N → V-set

**axiom**

82.  $\forall v:V \cdot v \in \text{obs\_Vs}(n) \Rightarrow$   
 82.  $\exists \text{onL}(li,fhi,f,thi):VP \cdot \text{onL}(li,fhi,f,thi)=\text{obs\_VP}(v) \Rightarrow$   
 82.  $\exists l:L \cdot l \in \text{obs\_Ls}(n) \wedge li=\text{obs\_LI}(l) \wedge \{fhi,thi\}=\text{obs\_HIs}(l) \vee$   
 82.  $\exists \text{atH}(hi):VP \cdot \text{atH}(hi)=\text{obs\_VP}(v) \Rightarrow$   
 82.  $\exists h:H \cdot h \in \text{obs\_Hs}(n) \wedge hi=\text{obs\_HI}(h)$

**3.3.2.1.1. • Static Wellformedness•**

85. We define a predicate over vehicle positions.

- (a) Every vehicle in the traffic has a proper position on the net, either at a hub or along a link.  
 (b) No two vehicles of the traffic can occupy exactly the same link position. (That is, the link positions  $\text{onL}(li,hi,f,hi')$  and  $\text{onL}(li,hi,f',hi')$  must have the two fractions  $(f, f')$  differ – be it ever so “minutely”).

We first define two auxiliary functions:<sup>2</sup>

**value**

- obs\_HIs: N → HI-set  
 obs\_HIs(n)  $\equiv \{\text{obs\_HI}(h) | h:H \cdot h \in \text{obs\_Hs}(n)\}$   
 obs\_LIs: N → LI-set  
 obs\_LIs(n)  $\equiv \{\text{obs\_LI}(h) | l:L \cdot l \in \text{obs\_Ls}(n)\}$

<sup>2</sup>They really ought to have been defined much earlier!

**3.3.2. Traffic**

83. By traffic we understand a continuous function from time to a pair of nets and position of vehicles.

84. By time we understand a dense set of points with dense and points being mathematical concepts [wayne.d.blizard.90,J.van.Benthem.Logic.Time91].

**type**

83. TF = T → (sel\_net:N × sel\_veh\_pos:(V → VP))  
 84. T

**3.3.2.1. Wellformedness of Traffic**

- Expressing the wellformedness of traffic is not a simple matter.
- We shall approach this task in a number of “small steps”.

85. proper\_vehicle\_positions: TF → Bool  
 85. proper\_vehicle\_positions(tf)  $\equiv$   
 85.  $\forall t:T \cdot t \in \text{DOMAIN } tf \cdot$   
 85. **let** (n,vps) = tf(t) **in**  
 85(a).  $\forall v:V \cdot v \in \text{dom } \text{vp\_is\_net\_position}(vps(v))(n)$   
 85(b).  $\forall v':V \cdot v' \in \text{dom } \text{vp} \wedge v \neq v' \Rightarrow \text{diff\_net\_pos}(vps(v),vps(v'))$   
 85. **end**  
 85(a). is\_net\_position: VP → N → Bool  
 85(a). is\_net\_position(vp)(n)  $\equiv$   
 85(a). **case** vp **of**  
 85(a). atH(hi) → hi ∈ obs\_HIs(n),  
 85(a). onL(li,fhi,f,thi) → li ∈ obs\_LIs(n) ∧ {fhi,thi} ⊆ obs\_HIs(n)  
 85(a). **end**  
 85(b). diff\_net\_pos: VP × VP → Bool  
 85(b). diff\_net\_pos(vp,vp')  $\equiv$   
 85(b). **case** (vp,vp') **of**  
 85(b). (atH(hi),atH(hi)) → true,  
 85(b). (onL(li,fhi,f,thi),onL(li,fhi,f',thi)) → f ≠ f',  
 85(b). \_ → true  
 85(b). **end**

### 3.3.2.1.2. • Dynamic Wellformedness•

86. Vehicles, when moving, move monotonically, that is,

- (a) if a vehicle, at some time,  $t$ , is at a link position  $\text{onL}(li,hi,f,hi')$  where  $f$  is not infinitesimally close to 1, then that vehicle will, at some later time  $t'$ , infinitesimally close to  $t$ , be at link position  $\text{onL}(li,hi,f',hi')$  where  $f'$  is infinitesimally close to  $f$ ;
- (b) if the vehicle, at some time,  $t$ , is at a link position  $\text{onL}(li,hi,f,hi')$  where  $f$  is indeed infinitesimally close to 1, then that vehicle will, at some infinitesimally later time  $t'$ , be at hub position  $\text{atH}(hi')$ ;
- (c) and if the vehicle, at some time,  $t$ , is at a hub position  $\text{atHP}(hi)$  then the vehicle will at some infinitesimally later time  $t'$  either be at hub position  $\text{atHP}(hi)$  or at some link position  $\text{onL}(li,hi,f,hi')$  where  $f$  is infinitesimally close to 0.

87. If a vehicle is (has been) moving along a link  $l_i$  and is now,

- at time  $t$ , at position  $\text{onL}(l_i, h_j, f, h_k)$ , that is, moving from  $h_j$  to  $h_k$ ,
- then it cannot at a subsequent, infinitesimally close time,  $t'$ , be at a position
- $\text{onL}(l_i, h_k, f', h_j)$ , that is, moving in the opposite direction,  $h_k$  to  $h_j$ .

**value**

```

86. monotonic: TF → Bool
86. monotonic(tf) ≡
86.   ∀ t,t':T · {t,t'} ⊆ DOMAIN tf ·
86.   let (n,vps) = tf(t),(n',vps')=tf(t') in
86.     INFINITESIMALLY CLOSE (t,t') ∧ t < t' ⇒
86.     ∀ v:V · v ∈ dom vps ∩ dom vps' ·
86.     case (vps(v),vps'(v)) of
86(a).       (onL(li,fhi,f,thi),onL(li,fhi,f',thi)) →
86(a).         f < f' ∧ INFINITESIMALLY CLOSE (f,f'),
86(b).       (onL(li,fhi,f,thi),atH(thi)) →
86(b).         INFINITESIMALLY CLOSE (f,1),
86(c).       (atH(hi),atH(hi)) → true,
86(c).       (atH(hi),onL(li,hi,f,thi)) →
86(c).         INFINITESIMALLY CLOSE (0,f),
86.         _ → true
86.   end end

```

**value**

```

87. God_does_not_play_dice3: TF → Bool
87. God_does_not_play_dice(tf) ≡
87.   ∀ t,t':T · {t,t'} ⊆ DOMAIN tf ∧ t < t' ∧ INFINITESIMALLY CLOSE (t,t') ⇒
87.   let (n,vps) = tf(t),(n',vps')=tf(t') in
87.     ∀ v:V · v ∈ dom vps ∩ dom vps' ⇒
87.     case (vps(v),vps'(v)) of
87.       (onL(li,fhi,_,thi),onL(li,thi,_,fhi)) → false,
87.       _ → true
87.   end end

```

<sup>3</sup>Albert Einstein: "I, at any rate, am convinced that He does not throw dice." Letter to Max Born (4 December 1926); The Born-Einstein Letters (translated by Irene Born) (Walker and Company, New York, 1971) ISBN 0-8027-0326-7. Reflects Einstein's view of Quantum Mechanics at the time.

88. If a vehicle is (has been) moving along and has,

- at time  $t$ , been at some position  $p$ , and
- at time  $t'$ , later than  $t$ , is at some position  $p'$ ,
- then it must at all times  $t''$  between  $t$  and  $t'$  have been somewhere on the net.

### value

88. no\_ghost\_vehicles:  $TF \rightarrow \mathbf{Bool}$

88. no\_ghost\_vehicles(tf)  $\equiv$

88.  $\forall t, t': T \cdot \{t, t'\} \subseteq \text{DOMAIN } tf \wedge t < t' \Rightarrow$

88. **let** (n,vps) = tf(t), (n',vps')=tf(t') **in**

88.  $\forall v: V \cdot v \in \mathbf{dom } vps \cap \mathbf{dom } vps' \Rightarrow$

88.  $\forall t'': T \cdot t < t'' < t' \Rightarrow$

88. **let** (n'',vps'') = tf(t'') **in**  $v \in \mathbf{dom } vps''$  **end**

88. **end**

### 3.3.4. Another Traffic Regulator

- We present an abstraction of a more conventional traffic signal than modelled in Items 74 on page 78 to 77 on page 81.

90. A traffic signal now simply shows an entry permit: either **red**, **yellow** or **green** at the hub when “leaving” any link, i.e., at the entry to a hub from any link.

### type

90.  $EP == \text{red} \mid \text{yellow} \mid \text{green}$

90.  $H\Sigma = LI \ \overline{m} \ EP$

### axiom

90.  $\forall h: H \cdot \text{obs\_LLs}(h) = \mathbf{dom } \text{obs\_H}\Sigma(h)$

- We leave it to the reader to express a constraint over hub state spaces as to how there must be hub states such that entry from any link is possible.

### 3.3.3. Traffic Rules (I of II)

89. A vehicle must not move from a hub,  $h_i$ , into a link  $\ell$  (from hub (identified by)  $h_i$  to hub (identified by)  $h_j$ ) which is closed in direction  $(h_i, h_j)$ , that is, where  $(h_i, h_j)$  is not in the current state of link.

#### rule:

89.  $\forall tf: TF, t: T \cdot t \in \text{DOMAIN}(tf) \Rightarrow$

89. **let** (n,tp) = tf(t) **in**

89.  $\forall v: V \cdot v \in \mathbf{dom } tp \Rightarrow$

89. **case** tp(v) **of**

89. atH(hi)  $\rightarrow$

89. **let**  $t': T \cdot t' > t \wedge t' \in \text{DOMAIN}(tr') \wedge \text{INFINITESIMALLY\_CLOSE}(t, t')$  **in**

89. **let** (n',tp') = tf(t') **in**

89.  $\exists li: LI, hi': HI, f: F, hi'': HI \cdot$

89.  $hi' = hi \wedge \text{INFINITESIMALLY\_CLOSE}(f, 0) \wedge$

89.  $tp'(v) = \text{onL}(li, hi', f, hi'') \wedge (hi, hi'') \notin \text{obs\_L}\Sigma(\text{getL}(li, n'))$

89.  $\_ \rightarrow \dots$

89. **end end end end**

### 3.3.5. Traffic Rules (II of II)

91. Vehicles must not enter a hub if entry permission is not **green**.

#### rule:

91.  $\forall tf: TF, t: T \cdot t \in \text{DOMAIN}(tf) \Rightarrow$

91. **let** (n,vps) = tf(t) **in**

91.  $\forall v: V \cdot v \in \mathbf{dom } vps \Rightarrow$

91. **case** vps(v) **of**

91. onL(li, hi, f, hi')  $\rightarrow$

91.  $\text{INFINITESIMALLY\_CLOSE}(f, 1) \wedge$

91. **let**  $h\sigma = \text{obs\_H}\Sigma(\text{getH}(hi', n)),$

91.  $t': T \cdot t' > t \wedge \text{INFINITESIMALLY\_CLOSE}(t, t')$  **in**

91. **let** (n',vps') = vps(t') **in**

91.  $h\sigma(li) \neq \text{green} \wedge vps'(v) \neq \text{atH}(hi')$  **assert:**  $vps'(v) = \text{onL}(li, hi, f, hi')$

91. **end end**

91.  $\_ \rightarrow \dots$

91. **end end**

## End of Lecture 3: DOMAINS: Intrinsic – Rules & Regulations

2.5.1.2 Wellformedness of Construction Plans	43
2.5.2 Augmented Construction Plans	47
2.5.3 Sequential Construction Behaviours	50
<b>Lect. #3: DOMAINS: Intrinsic – Rules &amp; Regulations</b>	<b>52</b>
<b>3 An Ontology of Domain Facets</b>	<b>53</b>
3.0.1 Definitions	53
3.0.2 What Can Be Observed	57
3.0.3 Business Processes	58
3.0.3.1 A Characterisation	58
3.0.3.2 An Example	58
3.1 Intrinsic	59
3.1.1 Net Topology Descriptors	60
3.1.2 Link States and Link State Spaces	63
3.1.3 Hub States and Hub State Spaces	67
3.1.4 State and State Space Wellformedness	69
3.1.5 Concrete Types for Simple Entities	70
3.1.6 Example Hub Crossings	73
3.1.7 Actions Continued	75
3.2 Support Technologies	77
3.2.1 Traffic Signals	78
3.2.2 Traffic “Control”	82
3.3 Rules and Regulations	84
3.3.1 Vehicles	87
3.3.2 Traffic	89
3.3.2.1 Wellformedness of Traffic	89
3.3.2.1.1 Static Wellformedness	90
3.3.2.1.2 Dynamic Wellformedness	92
3.3.3 Traffic Rules (I of II)	97
3.3.4 Another Traffic Regulator	98
3.3.5 Traffic Rules (II of II)	99

## B Slide Table-of-Contents

### Contents

<b>Lect. #1: COVER &amp; INTRODUCTION</b>	<b>0</b>
<b>1 Introduction</b>	<b>5</b>
1.1 The Problem	5
1.2 The Triptych Approach	6
<b>Lect. #2: ONTOLOGY</b>	<b>8</b>
<b>2 An Ontology of Specification Entities</b>	<b>9</b>
2.1 Simple Entities	11
2.1.1 Net, Hubs and Links	13
2.1.2 Unique Hub and Link Identifiers	14
2.1.3 Observability of Hub and Link Identifiers	15
2.1.4 A Theorem	17
2.1.4.1 Links implies Hubs	17
2.1.5 Hub and Link Attributes	18
2.1.6 Hub and Link Generators	19
2.2 States	22
2.3 Actions	22
2.3.1 Insert Hubs	23
2.3.2 Remove Hubs	25
2.3.3 Insert Links	27
2.3.4 Remove Links	31
2.3.5 Two Theorems	34
2.3.5.1 Idempotency	34
2.3.5.2 Reachability	35
2.4 Events	37
2.5 Behaviours	41
2.5.1 Behaviour Prescriptions	42
2.5.1.1 Construction Plans	42

<b>Lect. #4: DOMAINS: Scripts – Human Behaviour</b>	<b>98</b>
3.5 Scripts	99
3.5.1 Routes as Scripts	100
3.5.1.1 Paths	100
3.5.1.2 Routes	103
3.5.2 Bus Timetables as Scripts	106
3.5.2.1 Buses	106
3.5.2.2 Bus Stops	106
3.5.2.3 Bus Routes	107
3.5.2.4 Bus Schedule	109
3.5.2.5 Timetable	111
3.5.3 Route and Bus Timetable Denotations	114
3.5.4 Licenses and Contracts	116
3.5.4.1 Contracts	122
3.5.4.2 Contractual Actions	127
3.5.4.3 Wellformedness of Contractual Actions	130
3.6 Management and Organisation	136
3.6.1 Transport System Examples	140
3.7 Human Behaviour	143
3.8 Towards Theories of Domain Facets	145
3.8.1 A Theory of Intrinsic	146
3.8.2 Theories of Support Technologies	147
3.8.2.1 An Example	147
3.8.2.2 General	150
3.8.3 A Theory of Rules & Regulations	151
3.8.4 A Theory of Management & Organisation	161
3.8.5 A Theory of Human Behaviour	162
<b>Lect. #5: REQUIREMENTS – up to and incl. Determination</b>	<b>164</b>
<b>4 An Ontology of Requirements Constructions</b>	<b>165</b>
4.1 Business Process Re-engineering	168
4.1.1 The Kinds of Requirements	171
4.1.2 Goals Versus Requirements	172
4.1.2.1 Goals of a Toll Road System	174

4.1.2.2 Goals of Toll Road System Software	175
4.1.2.3 Arguing Goal-satisfaction of a Toll Road System	176
4.1.2.4 Arguing Goal-satisfaction of Toll Road System Software	177
4.1.3 Re-engineered Nets	179
4.2 Domain Requirements	190
4.2.1 Projection	192
4.2.1.1 Example	194
4.2.2 Instantiation	195
4.2.2.1 Example	196
4.2.2.2 Abstraction: From Concrete Toll Road Nets to Abstract Nets	201
4.2.2.3 Theorem	202
4.2.3 Determination	203
4.2.3.1 Example	204
<b>Lect. # 6: REQUIREMENTS – from Extension "out"</b>	<b>207</b>
4.2.4 Extension	208
4.2.4.1 Intuition	211
4.2.4.2 Descriptions	213
4.2.4.2.1 A RAISE/CSP Model	213
4.2.4.2.1 Toll Booth Plazas	213
4.2.4.2.1 Cars	215
4.2.4.2.1 Entry Booths	216
4.2.4.2.1 Gates	218
4.2.4.2.1 The Entry Plaza System	219
4.2.4.2.2 A Duration Calculus Model	224
4.2.4.2.3 A Timed Automata Model	228
4.2.5 Fitting	232
4.2.5.1 Examples	233
4.3 Interface Requirements	234
4.3.1 But First: On Shared Phenomena and Concepts	236
4.3.2 Shared Simple Entities	237
4.3.2.1 Example	238
4.3.3 Shared Actions	239
4.3.3.1 Example	240
4.3.4 Shared Events	241

1.1.2.2 Subtypes	281
Example 5: Net Subtypes	282
1.1.2.3 Sorts — Abstract Types	288
Example 6: Net Sorts	289
<b>Lect. # 8: RSL: Values &amp; Operations</b>	<b>289</b>
1.2 Concrete RSL Types: Values and Operations	290
1.2.1 Arithmetic	290
1.2.2 Set Expressions	291
1.2.2.1 Set Enumerations	291
Example 7: Set Expressions over Nets	292
1.2.2.2 Set Comprehension	296
Example 8: Set Comprehensions	297
1.2.3 Cartesian Expressions	298
1.2.3.1 Cartesian Enumerations	298
Example 9: Cartesian Net Types	299
1.2.4 List Expressions	302
1.2.4.1 List Enumerations	302
1.2.4.2 List Comprehension	303
Example 10: Routes in Nets	304
1.2.5 Map Expressions	309
1.2.5.1 Map Enumerations	309
1.2.5.2 Map Comprehension	310
Example 11: Concrete Net Type Construction	311
1.2.6 Set Operations	314
1.2.6.1 Set Operator Signatures	314
1.2.6.2 Set Examples	315
1.2.7 Cartesian Operations	316
1.2.8 List Operations	317
1.2.8.1 List Operator Signatures	317
1.2.8.2 List Operation Examples	318
1.2.9 Map Operations	319
1.2.9.1 Map Operator Signatures and Map Operation Examples	319

4.3.4.1 Examples	242
4.3.5 Shared Behaviours	243
4.3.5.1 Example	244
4.4 Machine Requirements	245
4.4.1 An Enumeration of Classes of Machine Requirements	246

**Lect. # 11: CLOSING****246****5 Conclusion****247**

5.1 What Have We Omitted	247
5.2 Domain Descriptions Are Not Normative	248
5.3 "Requirements Always Change"	249
5.4 What Can Be Described and Prescribed	251
5.5 What Have We Achieved – and What Not	253
5.6 Relation to Other Work	254
5.7 "Ideal" Versus Real Developments	257
5.8 Description Languages	259
5.9 Entailments	261
5.10 Domain Versus Ontology Engineering	262

**6 Bibliographical Notes****263**

6.1 Description Languages	263
---------------------------	-----

**Lect. # 7: RSL: Types****264****1 An RSL Primer****265**

1.1 Types	265
1.1.1 Type Expressions	265
1.1.1.1 Atomic Types	265
Example 1: Basic Net Attributes	267
1.1.1.2 Composite Types	269
Example 2: Composite Net Type Expressions	270
1.1.2 Type Definitions	272
1.1.2.1 Concrete Types	272
Example 3: Composite Net Types	273
Example 4: Net Record Types: Insert Links	279

**Lect. # 9: RSL: Logic,  $\lambda$ -Calculus, Fctl. Specs.****320**

1.3 The RSL Predicate Calculus	321
1.3.1 Propositional Expressions	321
1.3.2 Simple Predicate Expressions	322
1.3.3 Quantified Expressions	323
Example 12: Predicates Over Net Quantities	324
1.4 $\lambda$ -Calculus + Functions	327
1.4.1 The $\lambda$ -Calculus Syntax	327
1.4.2 Free and Bound Variables	328
1.4.3 Substitution	329
1.4.4 $\alpha$ -Renaming and $\beta$ -Reduction	330
Example 13: Network Traffic	331
1.4.5 Function Signatures	338
Example 14: Hub and Link Observers	339
1.4.6 Function Definitions	341
Example 15: Axioms over Hubs, Links and Their Observers	344
1.5 Other Applicative Expressions	345
1.5.1 Simple Iet Expressions	345
1.5.2 Recursive Iet Expressions	346
1.5.3 Non-deterministic Iet Clause	347
1.5.4 Pattern and "Wild Card" Iet Expressions	348
1.5.5 Conditionals	349
Example 16: Choice Pattern Case Expressions: Insert Links	350
1.5.6 Operator/Operand Expressions	360

**Lect. # 10: RSL: Imperative & Process Specs.****360**

1.6 Imperative Constructs	361
1.6.1 Statements and State Changes	361
1.6.2 Variables and Assignment	362
1.6.3 Statement Sequences and skip	362
1.6.4 Imperative Conditionals	362
1.6.5 Iterative Conditionals	363
1.6.6 Iterative Sequencing	363
1.7 Process Constructs	364
1.7.1 Process Channels	364

	<b>Example 17: Modelling Connected Links and Hubs</b> . . . . .	365
1.7.2 Process	Definitions . . . . .	369
	<b>Example 18: Communicating Hubs, Links and Vehicles</b> . . . . .	371
1.7.3 Process	Composition . . . . .	373
	<b>Example 19: Modelling Transport Nets</b> . . . . .	374
1.7.4 Input/Output	Events . . . . .	377
	<b>Example 20: Modelling Vehicle Movements</b> . . . . .	378
1.8 Simple RSL	Specifications . . . . .	384
	<b>Example 21: A Neat Little "System"</b> . . . . .	388
<b>B Slide Table-of-Contents</b>		<b>397</b>