3 An Ontology of Domain Facets 53

Definition: Domain. *An area of activity which some software[685] 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****.*
Usually a domain description is a set of documents with many parts recording many facets of the domain: The *business process*[99]es, *intrinsics*[399], *support technology*[725], *rules and regulations*[640], *management and organisation*[445], and the *human behaviour*[345]s.
Definition: Domain Engineering. *The engineering of the development of a domain description[243], from identification of domain[239] stakeholder[703]s, via domain acquisition[240], domain analysis[241], terminologisation, and domain description[243] to domain validation[256] and domain verification[257].*
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 process*[99]es, *intrinsics*[399], *support technology*[725], *rules and regulations*[640], *management and organisation*[445], and *human behaviour*[345].

3.1 What Can Be Observed 57

- "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.

3.2 Intrinsics 58

Definition: Intrinsics. *By the intrinsics of a domain[239] we shall understand those phenomena and concepts of a domain which are basic to any of the other facets, with such a domain intrinsics initially covering at least one stakeholder[703] view.*

3.2.1 Net Topology Descriptors 59

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.

52. 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,
53. such that
    a) for all hi in (the definition set of) nd it is the case that
    b) if hi maps to lihi,
    c) and in that link identifier to hub identifier map, li maps to hi′,
    d) then hi′ is different from hi and
    e) hi′ maps to an lihi′ in which li is defined and maps to hi.
    f) And there are only such pairings.

60

```
type
52.  ND′ = HI m→ (LI m→ HI)
52.  ND = {|nd′:ND•wf_ND(nd′)|}
value
53.  wf_ND: ND′ → Bool
53.  wf_ND(nd) ≡
53a.     ∀ hi:HI•hi ∈ dom nd ⇒
53b.       let lihi = nd(hi) in
53c.       ∀ li:LI • li ∈ dom lihi ⇒
53c.         let hi′ = (nd(hi))(li) in
53d.         hi ≠ hi′ ∧
53e.         hi′ ∈ dom nd ∧ li ∈ dom(nd(hi′)) ∧ hi=(nd(hi′))(li)
53f.       end end
```

61

From a net one can construct its net descriptor:

```
value
  conND: N → ND
  conND(n) ≡
      [ hi↦[ li↦hi′|li:LI,hi′:HI•li ∈ obs_LIs(getH(hi,n))∧{hi,hi′}=obs_HIs(getL(li,n)) ]|
          hi:HI•hi ∈ xtrHIs(n) ]
```

3.2.2 Link States and Link State Spaces

62

Links are (one of the) means of transport[6]. Hubs allow movement along one (hub-connected) link to be diverted onto another (hub-connected) link.

[6] Other means are vehicles moving along links and crossing hubs and the locomotive force that drives the vehicles. Freight, including people, are what is being transported.

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
63 the hub to potentially any link.

Let the observed hub identifiers of a link ℓ be $\{h_j, h_k\}$, then link ℓ 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
64 time undergo transitions between any subset of the potential link state space.

54. Link states, $l\sigma{:}L\Sigma$, are set of pairs of hub identifiers.
55. Link state spaces are set of link states.
56. From a link one can generate the link state space of all potential link states.
57. From a link one can observe the current link state $l\sigma{:}L\Sigma$.
58. From a link one can observe the link state space $l\omega{:}L\Omega$.

65

type
54. LΣ = (HI×HI)**-set**
55. LΩ = LΣ**-set**

value
56. generate_full_LΣ: L → LΣ
56. generate_full_LΣ(l) ≡
56. {}∪{(hi′,hi″)|hi′,hi″:HI•hi′≠hi″∧{hi′,hi″}=obs_HIs(l)}

56. generate_LΩ: L → LΩ
56. **let** fullLσ = generate_full_LΣ(l) **in**
56. {{},∪{σ|σ:LΣ•$\sigma\subseteq$fullLσ}} **end**

57. obs_LΣ: L → LΣ
58. obs_LΩ: L → LΣ**-set**

3.2.3 Hub States and Hub State Spaces 66

59. 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.
60. Hub state spaces are set of hub states.
61. From a hub one can generate the hub state space of all potential hub states.

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

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

67

type
59. HΣ = (LI×LI)-**set**
60. HΩ = HΣ-**set**
value
61. generate_full_HΣ: H → HΣ
61. generate_full_HΣ(h) ≡
61. {}∪{(li′,li″)|li′,li″:LI•{li′,li″}⊆obs_LIs(h)}

56. generate_HΩ: H → HΩ
56. **let** fullHσ = generate_full_HΣ(h) **in**
56. {{}∪{σ|σ:HΣ•σ⊆fullHσ}} **end**

62. obs_HΣ: H → HΣ
62. obs_HΩ: H → HΣ-**set**

3.2.4 State and State Space Wellformedness 68

64. States must be in appropriate state spaces.

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

axiom
∀ n:N,l:L,h:H • l ∈ obs_Ls(n) ∧ h ∈ obs_Hs(n) ⇒
54. obs_LΣ(l) ∈ obs_LΩ(l) ∧
55. obs_LΩ(l) ⊆ generate_full_LΣ(l) ∧
54. obs_HΣ(h) ∈ obs_HΩ(h) ∧
55. obs_HΩ(h) ⊆ generate_full_HΣ(h)

theorems:
∀ n:N,l:L,h:H • l ∈ obs_Ls(n) ∧ h ∈ obs_Hs(n) ⇒
obs_LΣ(l) ⊆ {(hi′,hi″)|hi′,hi″:H•{hi′,hi″}⊆obs_HIs(l)} ∧
obs_HΣ(h) ⊆ {(li′,li″)|li′,li″:L•{li′,li″}⊆obs_LIs(h)}

3.2.5 Concrete Types for Simple Entities 69

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.

66. Nets are Cartesians of sets of hubs and links.
67. 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.
68. 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.

70

type
66. N = H-**set** × L-**set**
67. L :: obs_LI:LI × obs_HIs:HI-**set** × LΣ × LΩ × LAtrs
68. H :: obs_HI:HI × obs_LIs:LI-**set** × HΣ × HΩ × HAtrs

71

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

axiom
∀ (hs,ls):N • ls≠{} ⇒ **card** hs ≥ 2 ∧
  ∀ l′,l″:L • {l′,l″}⊆ls ∧ l′≠l″ ⇒ obs_LI(l′)≠obs_LI(l″) ∧
  ∀ h′,h″:H • {h′,h″}⊆hs ∧ h′≠h″ ⇒ obs_HI(h′)≠obs_HI(h″) ∧
  ∀ l:(li,his,lσ,lω,latrs):L • l ∈ ls ⇒
    card his=2 ∧ his⊆{obs_HI(h″)|h‴:H • h‴ ∈ hs} ∧
    lσ ∈ generate_full_LΣ(l) ∧
    lσ ∈ lω ⊆ generate_full_LΣ(l) ∧
  ∀ h:(hi,lis,hσ,hω,hatrs):H • h ∈ hs ⇒
    lis⊆{obs_LI(l‴)|l‴:L • l‴ ∈ ls} ∧
    hσ ∈ generate_full_HΣ(h) ∧
    hσ ∈ hω ⊆ generate_full_HΣ(h)

3.2.6 Example Hub Crossings 72

Figure 1 shows four hub/partial link corner diagrams (1.–4.). These are intended to show four distinct hub states. Let the center diagram (5.) of Fig. 1 indicate the link identifiers 73 of the four partial links of each of the four hub/partial link diagrams.

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 on page 32 shows a semaphore which seems to be able to display all kinds of states.

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.

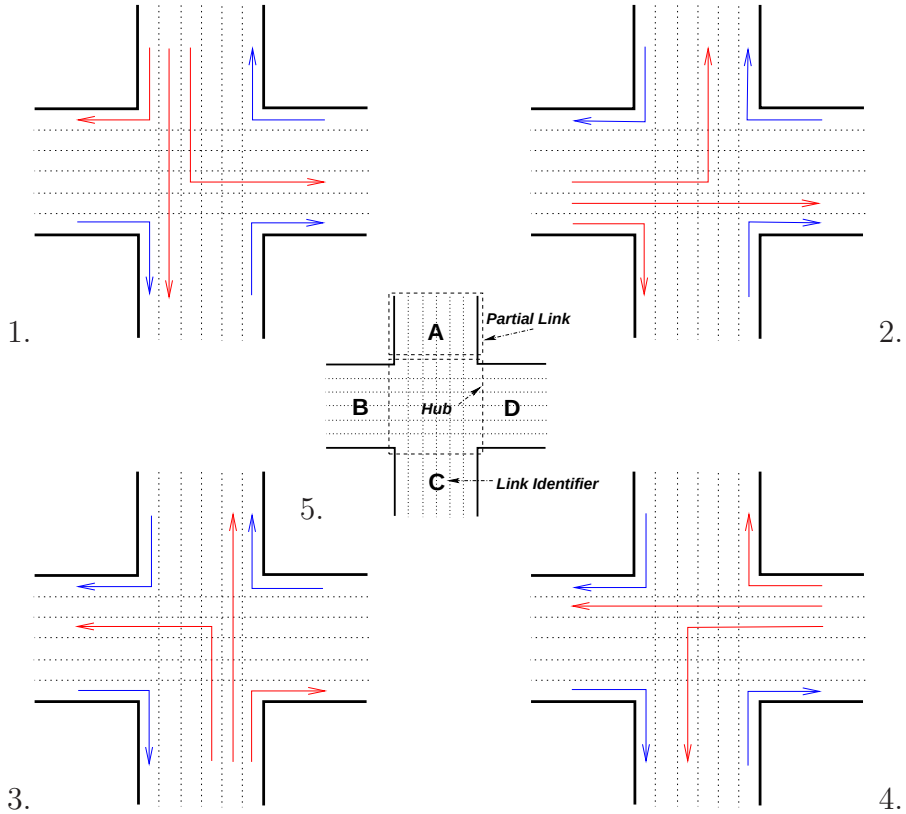

Figure 1: Four "Safe" Flows

3.2.7 Actions Continued 74

69. The action change_HΣ 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.

69b. 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.

75

value

69. change_HΣ: H × HΣ → H

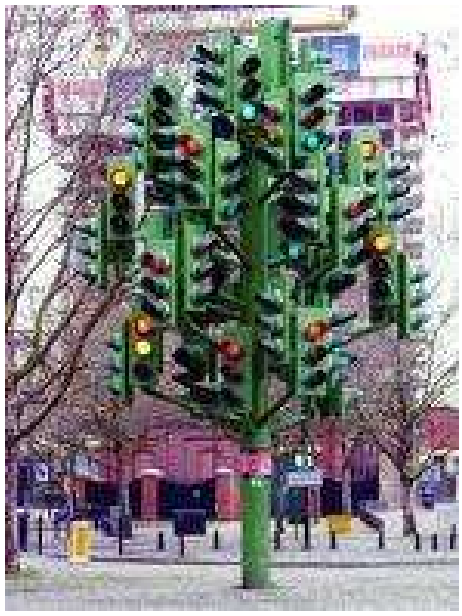

Figure 2: A General Purpose Traffic Light

69. change_HΣ((hi,lis,hσ,hω,hatrs),hσ′) ≡
69b. **let** hσ‴ ∈ generate_full_HΣs **in**
69a. (hi,lis,hσ‴,hω,hatrs) **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.

3.3 Support Technologies 76

Definition: Support Technology. *By a support technology we understand a facet[285] of a domain[239], 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 process[99] es.*

3.3.1 Traffic Signals 77

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

70. 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").

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

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

78

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 σ_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 $\{h\sigma_1, h\sigma_2, ..., h\sigma_{n-1}\}$ of

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

are called intermediate states.

Their purpose is to secure an orderly phase out of green via yellow to red and phase in of red via yellow to green in some order for the various directions.

We leave it to the reader to devise proper wellformedness conditions for signaling sequences as they depend on the hub topology. 79

72. 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
70. HΣ = LI × Colour × LI
70. Colour == red | yellow | green
71. Signalling = (HΣ × TI)*
71. TI
72. Sempahore = (HΣ×HΣ) $\xrightarrow[m]{}$ Signalling

value
72. obs_Semaphore: H → Sempahore

80

73. 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 semaphore 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.

73. hub_state_space: Sempahore → HΣ-**set**
73. hub_state_space(sema) ≡
73a. **let** hsps={hsp|hsp:(HΣ×HΣ)•hsp ∈ **dom** sema} **in**
73b. **let** hs={hσ′,hσ″|hσ′,hσ″:HΣ•(hσ′,hσ″)∈ hsps} **in**
73c. hs ∪ ∪{{hσ|(hσ,ti):(HΣ×TI)•(hσ,ti)∈ **elems** sg}|sg:Signalling•sg ∈ **rng** sema}
73. **end end**
axiom
73. ∀ h:H • ∪ obs_HΩ(h) = hub_state_space(obs_Semaphore(h))

3.3.2 Traffic "Control" 81

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

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

value
74. signalling: HΣ × HΣ → H → H
75. signalling(hσ$_{\text{init}}$,hσ$_{\text{end}}$)(h) ≡
75. **let** sema = obs_Semaphore(h) **in**
75. **let** sg = sema(hσ$_{\text{init}}$,hσ$_{\text{end}}$) **in**
75. signal_sequence(sg)(h) **end end**
75. **pre** (hσ$_{\text{init}}$,hσ$_{\text{end}}$) ∈ **dom** obs_Semaphore(h)

82

75. signal_sequence(⟨⟩)(h) ≡ h
75. signal_sequence(⟨(hσ,ti)⟩⌢sg)(h) ≡
75. **let** hσ′ = change_HΣ(h)(hσ) **in**
75. **if** hσ′ ≠ hσ **then chaos**
75. **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.

3.4 Rules and Regulations 83

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[536] *over a pair of suitably chosen domain* state[705]*s. We may assume that a domain* action[12] *or a domain* event[281] *takes place in the* first *of these* state*s and results in the* second *of these* state*s. If the* predicate *is true then we say that the rule has been* obeyed*, otherwise that it* 84 *has been* violated*.*
Usually a domain rule is paired with a possibly remedying regulation.

Definition: Regulation. *A regulation stipulates that an* ***action***[12] *be taken in order to remedy a previous action which violated a* ***rule***[638]*. That is, a regulation is some* ***text*** *which designates a possibly composite* ***action***[12]*, that is, a* ***state-to-state change*** *which ostensibly results in a state in which the* ***rule****, "attached" to the regulation,* ***now holds****.*

3.4.1 Vehicles 85

In preparation for examples of transportation rules and regulations we introduce vehicles.

76. 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.
77. From a net (sort) one can observe all the vehicles of the net.[7]
78. No two vehicles so observed have the same identifier.

86

type
76. V
76a. VI
76b. VP = HP | LP
76(b)i. HP == atH(hi:HI)
76(b)ii. LP == onL(li:LI,fhi:HI,f:F,thi:HI)
76(b)ii. F = {|f:F•0<f<1|}
value
76a. obs_VI: V → VI
76b. obs_VP: V → VP
77. obs_Vs: N → V-**set**
axiom
78. ∀ v:V • v ∈ obs_Vs(n) ⇒
78. ∃ onL(li,fhi,f,thi):VP • onL(li,fhi,f,thi)=obs_VP(v) ⇒
78. ∃ l:L•l ∈ obs_Ls(n)∧li=obs_LI(l)∧{fhi,thi}=obs_HIs(l) ∨
78. ∃ atH(hi):VP • atH(hi)=obs_VP(v) ⇒
78. ∃ h:H•h ∈ obs_Hs(n)∧hi=obs_HI(h)

MORE TO COME

[7] Thus a concrete net type, in addition to hubs and links (now) also contains vehicles.

3.4.2 Traffic 87

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

80. By time we understand a dense set of points with dense and points being mathematical concepts [57, 222].

type
79. TF = T → (sel_net:N × sel_veh_pos:(V $\xrightarrow[m]{}$ VP))
80. T

Wellformedness of Traffic

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

88

• Static Wellformedness

81. 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 onL(li,hi,f,hi′) and onL(li,hi,f′,hi′) must have the two fractions (f, f') differ – be it ever so "minutely").

We first define two auxiliary functions:[8]

value
obs_HIs: N → HI-**set**
obs_HIs(n) ≡ {obs_HI(h)|h:H•h ∈ obs_Hs(n)}
obs_LIs: N → LI-**set**
obs_LIs(n) ≡ {obs_LI(h)|l:L•l ∈ obs_Ls(n)}

89

81. proper_vehicle_positions: TF → **Bool**
81. proper_vehicle_positions(tf) ≡
81. ∀ t:T • t ∈ 𝔻𝕆𝕄𝔸𝕀ℕ tf •
81. **let** (n,vps) = tf(t) **in**
81a. ∀ v:V•v ∈ **dom** vp•is_net_position(vps(v))(n)
81b. ∀ v′:V•v′ ∈ **dom** vp ∧ v≠v′⇒diff_net_pos(vps(v),vps(v′))
81. **end**

[8] They really ought to have been defined much earlier!

```
81a. is_net_position: VP → N → Bool
81a. is_net_position(vp)(n) ≡
81a.   case vp of
81a.     atH(hi) → hi ∈ obs_HIs(n),
81a.     onL(li,fhi,f,thi) → li ∈ obs_LIs(n)∧{fhi,thi}⊆obs_HIs(n)
81a.   end
81b. diff_net_pos: VP × VP → Bool
81b. diff_net_pos(vp,vp′) ≡
81b.   case (vp,vp′) of
81b.     (atH(hi),atH(hi)) → true,
81b.     (onL(li,fhi,f,thi),onL(li,fhi,f′,thi)) → f≠f′,
81b.     _ → true
81b.   end
```

90

• Dynamic Wellformedness

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

    a) if a vehicle, at some time, t, is at a link position 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 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 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 atH(hi′);

    c) and if the vehicle, at some time, t, is at a hub position atHP(hi) then the vehicle will at some infinitesimally later time t' either be at hub position atHP(hi) or at some link position onL(li,hi,f,hi′) where f is infinitesimally close to 0.

91

```
value
82. monotonic: TF → Bool
82. monotonic(tf) ≡
82.   ∀ t,t′:T • {t,t′}⊆𝔻𝕆𝕄𝔸𝕀ℕ tf •
82.     let (n,vps) = tf(t),(n′,vps′)=tf(t′) in
82.     ℐℕℱℐℕ𝕀𝕋𝔼𝕊𝕀𝕄𝔸𝕃𝕃𝕐 ℂ𝕃𝕆𝕊𝔼 (t,t′)∧t<t′⇒
82.       ∀ v:V•v ∈ dom vps ∩ dom vps′ •
82.         case (vps(v),vps′(v)) of
82a.           (onL(li,fhi,f,thi),onL(li,fhi,f′,thi)) →
82a.             f<f′ ∧ ℐℕℱℐℕ𝕀𝕋𝔼𝕊𝕀𝕄𝔸𝕃𝕃𝕐 ℂ𝕃𝕆𝕊𝔼 (f,f′),
82b.           (onL(li,fhi,f,thi),atH(thi)) →
82b.             ℐℕℱℐℕ𝕀𝕋𝔼𝕊𝕀𝕄𝔸𝕃𝕃𝕐 ℂ𝕃𝕆𝕊𝔼 (f,1),
```

```
82c.              (atH(hi),atH(hi)) → true,
82c.              (atH(hi),onL(li,hi,f,thi)) →
82c.                  INFINITESIMALLY CLOSE (0,f),
82.               _ → true
82.       end   end
```

83. If a vehicle is (has been) moving along a link l_i and is now,
    - at time t, at position $\mathsf{onL}(l_i, h_j, f, h_k)$, that is, moving from h_j to h_k,
    - then it cannot at a subsequent, infinitesimally close time, t', be at a position
    - $\mathsf{onL}(l_i, h_k, f', h_j)$, that is, moving in the opposite direction, h_k to h_j.

```
value
83.  God_does_not_play_dice[9]: TF → Bool
83.  God_does_not_play_dice(tf) ≡
83.    ∀ t,t′:T • {t,t′}⊆DOMAIN tf ∧ t<t′ ∧ INFINITESIMALLY CLOSE (t,t′)⇒
83.      let (n,vps) = tf(t),(n′,vps′)=tf(t′) in
83.      ∀ v:V • v ∈ dom vps ∩ dom vps′ ⇒
83.        case (vps(v),vps′(v)) of
83.          (onL(li,fhi,_,thi),onL(li,thi,_,fhi))→false,
83.          _ → true
83.      end end
```

84. 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
84.  no_ghost_vehicles: TF → Bool
84.  no_ghost_vehicles(tf) ≡
84.    ∀ t,t′:T • {t,t′}⊆DOMAIN tf ∧ t<t′ ⇒
84.      let (n,vps) = tf(t),(n′,vps′)=tf(t′) in
84.      ∀ v:V•v ∈ dom vps ∩ dom vps′ ⇒
84.        ∀ t″:T • t<t″<t′ ⇒
84.          let (n″,vps″) = tf(t″) in v ∈ dom vps″ end
84.      end
```

[9] 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.

3.4.3 Traffic Rules (I of II) 95

85. A vehicle must not move from a hub, h_i, into a link ℓ (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:
85. ∀ tf:TF,t:T • t ∈ 𝔻𝕆𝕄𝔸𝕀ℕ(tf) ⇒
85. **let** (n,tp) = tf(t) **in**
85. ∀ v:V • v ∈ **dom** tp ⇒
85. **case** tp(v) **of**
85. atH(hi) →
85. **let** t′:T • t′>t ∧ t′ ∈ 𝔻𝕆𝕄𝔸𝕀ℕ(tr′) ∧ 𝕀ℕ𝔽𝕀ℕ𝕀𝕋𝔼𝕊𝕀𝕄𝔸𝕃𝕃𝕐_ℂ𝕃𝕆𝕊𝔼(t,t′) **in**
85. **let** (n′,tp′) = tf(t′) **in**
85. ∃ li:LI,hi′:HI,f:F,hi″:HI •
85. hi′=hi ∧ 𝕀ℕ𝔽𝕀ℕ𝕀𝕋𝔼𝕀𝕄𝔸𝕃𝕃𝕐_ℂ𝕃𝕆𝕊𝔼(f,0) ∧
85. tp′(v) = onL(li,hi′,f′,hi″) ∧(hi,hi″) ∉ obs_LΣ(getL(li,n′))
85. _ → ...
85. **end end end end**

We shall give another rule after the next section.

3.4.4 Another Traffic Regulator 96

We present an abstraction of a more conventional traffic signal than modelled in Items 70 on page 32 to 73 on page 33.

86. 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
86. EP == red | yellow | green
86. HΣ = LI $\xrightarrow[m]{}$ EP
axiom
86. ∀ h:H • obs_LIs(h)=**dom** obs_HΣ(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.

3.4.5 Traffic Rules (II of II) 97

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

rule:
87. ∀ tf:TF,t:T : t ∈ 𝔻𝕆𝕄𝔸𝕀ℕ(tf) ⇒
87. **let** (n,vps) = tf(t) **in**
87. ∀ v:V • v ∈ **dom** vps ⇒
87. **case** vps(v) **of**
87. onL(li,hi,f,hi′) →
87. 𝕀ℕ𝔽𝕀ℕ𝕀𝕋𝔼𝕊𝕀𝕄𝔸𝕃𝕃𝕐_ℂ𝕃𝕆𝕊𝔼(f,1) ∧
87. **let** hσ = obs_HΣ(getH(hi′,n)),
87. t′:T • t′>t ∧ 𝕀ℕ𝔽𝕀ℕ𝕀𝕋𝔼𝕊𝕀𝕄𝔸𝕃𝕃𝕐_ℂ𝕃𝕆𝕊𝔼(t,t′) **in**
87. **let** (n′,vps′) = vps(t′) **in**
87. hσ(li) ≠ green ∧ vps′(v) ≠ atH(hi′) **assert:** vps′(v) = onL(li,hi,f,hi′)
87. **end end**
87. _ → ...
87. **end end**

3.5 Scripts 98

Definition: Scripts. *A script is plan of action. By a domain script we shall, more specifically, understand the structured, almost, if not outright, formally expressed, wording of a set of rules and regulations*[640].
See also *license*[424] and *contract*[181]. Definitions follow.

3.5.1 Routes as Scripts 99

Paths

88. A path is a triple:
    a) a hub identifier, h_i, a link identifier, l_j, and another hub identifier, h_k, distinct from h_i,
    b) such that there is a link ℓ with identifier l_j in a net n such that $\{h_i, h_k\}$ are the hub identifiers that can be observed from ℓ.

type
88. Pth = HI × LI × HI
axiom
88a. ∀ (hi,li,hi′):Pth • ∃ n:N,l:L • l ∈ obs_Ls(n) ⇒
88b. obs_LI(l)=li ∧ obs_HIs(l)={hi,hi′}

100

89. From a net one can extract all its paths:
    a) if l is a link of the net,

b) l_j its identifier,

c) $\{h_i, h_k\}$ the identifiers of its connected hubs,

d) then (h_i, l_j, h_k) and (h_k, l_j, h_j) are paths of the net.

value
89. paths: N → Pth-**set**
89a. paths(n) ≡
89d. {(hi,lj,hk),(hk,lj,hi)|l:L,lj:LI,hi,hk:HI•l ∈ obs_Ls(n) ∧
89b. lj=obs_LI(l) ∧
89c. {hi,hk}=obs_HIs(l)}

101

90. From a net descriptor one can (likewise) extract all its paths:

a) Let h_i, h_k be any two distinct hub identifiers of the net descriptor (definition set),

b) such that they both map into a link identifier l_j,

c) then (h_i, l_j, h_k) and (h_k, l_j, h_j) are paths of the net.

value
89. paths: ND → Pth-**set**
89. paths(nd) ≡
90a. {(hi,lj,hk),(hk,lj,hi)|hi,hk:HI,lj:LI • hi≠hk ∧ {hi,hk}⊆**dom** nd ⇒
90b. lj ∈ **dom** nd(hi)∩ **dom** nd(hk)}

Routes

102

91. A route of a net is a sequence of zero, one or more paths such that

a) all paths of a route are paths of the net and

b) adjacent paths in the sequence "share" hub identifiers.

type
91. R = Pth*
axiom
91. ∀ r:R, ∃ n:N •
91a. **elems** r ⊆ paths(n) ∧
91b. ∀ i:**Nat** • {i,i+1}⊆**inds** r ⇒
91b. **let** (_,_,hi)=r(i), (hi′,_,_)=r(i+1) **in** hi=hi′ **end**

103

92. From a net, n, we can generate the possibly infinite set of finite and possibly infinite routes:
    a) <> is a path (***basis clause 1***);
    b) if p is a path of n then $< p >$ is a path of n (***basis clause 2***);
    c) if r and r' are non-empty routes of n
        i. and the last h_i of r is the same as the first h_j of r'
        ii. then the concatenation of r and r' is a route

    (***induction clause***).
    d) Only such routes which can be formed by a (finite, respectively infinite) application of basis clauses Items 92a and 92b and induction clause Item 92c are routes (***extremal clause***).

104

```
value
92.  routes: N|ND → R-infset
92.  routes(nond) ≡
92a.   let rs = {⟨⟩} ∪
92b.           {⟨p⟩|p:Pth•p ∈ paths(nond)} ∪
92(c)ii.          {r⌢r′|r,r′:R • r ∈ rs ∧ r′ ∈ rs ∧
92(c)i.                 ∃ hi,hi′,hi″,hi‴:H,li:LI •
92(c)i.                 r=r″⌢⟨(hi,li,hi′)⟩∧r′=⟨(hi″,li′,hi‴)⟩⌢r‴ ∧
92(c)i.                 hi′=hi″} in
92d.   rs end
```

3.5.2 Bus Timetables as Scripts

105

Buses

93. Buses are vehicles,
94. with bus identifiers being the same as vehicle identifiers.

```
type
93.  B
94.  BI ⊆ VI
```

Bus Stops

95. A link bus stop indicates the link (by its identifier), the from and to hub identifiers, and the fraction "down the link" from the from to the to hub identifiers.

```
type
95.  BS = mkL_BS(sel_fhi:HI,sel_li:LI,sel_f:F,sel_thi:HI)
```

Bus Routes 106

96. A bus stop list is a sequence of two or more bus stops, *bsl*.

97. A bus route, *br*, is a pair of a net route, *r*, and a bus stop list , *bsl*, such that route *r* is a route of *n* and such that *bsl* is embedded in *r*. If

    a) there exists an index list, *il*, of ascending indices of the route *r* and of the length of *bsl*

    b) such that the *i*th path of *r*

    c) share from and to hub identifiers and link identifier with the *il*(*i*)th bus stop of *bsl*

    then *bsl* is embedded in *r*.

98. We must allow for two or more stops along a bus route to be adjacent on the same link — in which case the corresponding fractions must likewise be ascending.

107

```
value
  n:N
type
96.  BSL = BS*
97.  BR = {|(r,bsl):(R×BSL)•wf_BR(r,bsl)|}
value
97.  wf_BR: BR → Bool
97.  wf_BR(r,bsl) ≡ ∃ n:N,r:R•r ∈ routes(n) ∧ is_embedded_in(r,bsl)

97a. is_embedded_in: BR → Bool
97a. is_embedded_in(r,bsl) ≡
97b.   ∃ il:Nat* • len il=len bsl∧inds il⊆inds r∧ascending(il) ⇒
97c.     ∀ i:Nat • i ∈ inds il ⇒
97c.       let (hi,lj,hk) = r(il(i)),(hi′,lj′,f,hk′) = bsl(i) in
97c.       hi=hi′ ∧ lj=lj′ ∧ hk=hk′ end ∧
98.      ∀ i:Nat • {i,i+1}⊆inds il ⇒
98.        let (hi,lj,f,hk)=bsl(i),(hi′,lj′,f′,hk′)=bsl(i+1) in
98.        hi=hi′ ∧ lj=lj′ ∧ hk=hk′ ⇒ f<f′ end

    ascending: Nat* → Bool, ascending(il) ≡ ∀ i:Nat•{i,i+1}⊆inds il ⇒ il(i)≤il(i+1)
```

The ≤ of the ascending predicate allows for more than one stop along the same route

Bus Schedule 108

99. A timed bus stop is a pair of a time and a bus stop.
100. A timed bus stop list is a sequence of timed bus stops.
101. A bus schedule is a pair of a route and a timed bus stop list such that
    - there is a net of which the routes is indeed a route,
    - the bus stop list of the timed bus stop list is embedded in the route, and
    - 'later" listed bus stops register later times.
102. SimpleBusSchedules remove routes from BusRoutes.

109

type
99. TBS :: sel_T:T sel_bs:BS
100. TBSL = TBS*
101. BusSched = {|(r,tbsl):(R×TBSL)•wf_BusSched(r,tbsl)|}

value
101. wf_BusSched: BusSched → **Bool**
101. wf_BusSched(r,tbsl) ≡
101. ∃ n:N•r ∈ routes(n)
101. ∧ **let** bsl:SBS = ⟨sel_BS(tbsl(i))|i:[1..**len** tbsl]⟩ **in** is_embedded_in(r,bsl) **end**
101. ∧ ∀ i:**Nat**•{i,i+1}⊆**inds** tbsl ⇒ sel_T(tbsl(i))<sel_T(tbsl(i+1))

type
102. SBS = {|bsl:BS*•∃ n:N,r:R•r ∈ routes(n)∧is_embedded_in(r,bsl)|}

Timetable 110

The concept of a bus line captures all those bus schedules which ply the same bus route but at different times. A timetable is made up from distinctly named bus lines.

103. A bus line has a unique bus line name.
104. We say that two bus schedules are the same if they are based on the same route and if they differ only in their times.
105. Each of the different bus routes of a bus line has a unique bus number.
106. A route bus schedule pairs a route with simple bus schedules for each of a number of busses (identified by their bus number).
107. A bus timetable (listing, map) maps bus line names to route bus schedules.
108. A timetable is a pair, a net and a table.

109. A well-formed timetable must satisfy same bus schedules within each bus line

110. All bus numbers are distinct across bus lines.

111

type
103. BLNm
value
104. same_bus_schedule: BusSched × BusSched → **Bool**
104. same_bus_schedule((r1,btl1),(r2,btl2)) ≡
104. r1 = r2 ∧ **len** btl1 = btl2 ∧
104. ⟨sel_BS(btl1(i))|i:[1..**len** btl1]⟩=⟨sel_BS(btl2(i))|i:[1..**len** btl2]⟩
type
105. BNo
106. RBS :: sel_R:R sel_btbl:(BNo $\overrightarrow{m}$ SBS)
107. TBL = BLNm $\overrightarrow{m}$ RBS
108. TT′ = ND × TBL
109. TT = {|tt:TT′•wf_TT(tt)|}

112

value
109. wf_TT: TT′ → **Bool**
109. wf_TT(_,tbl) ≡
109. ∀ bln:BLNm•bln ∈ **dom** tbl ⇒
109. ∀ bno,bno′:BNo • {bno,bno′}⊆**dom** sel_btbl(tbl(bln)) ⇒
109. same_bus_schedule(sel_R(tbl(bln)),sel_btbl(tbl(bln))(bno),
109. sel_R(tbl(bln)),sel_btbl(tbl(bln))(bno′)) ∧
110. ∀ bln′,bln″:BLNm • {bln′,bln″}⊆**dom** tbl ∧ bln′≠bln″ ⇒
110. **dom** sel_btbl(tbl(bln′)) ∩ **dom** sel_btbl(tbl(bln″)) = {}

3.5.3 Route and Bus Timetable Denotations 113

What are routes and bus timetables scripting ?

Routes (list of connected link traversal designations) script that one may transport people or freight along the sequence of designated links.

Bus timetables script (at least) two things: the set of bus traffics on the net which satisfy the bus timetable, and information that potential and actual bus passengers may, within some measure of statistics (and probability), rely upon for their bus transport. 114

Here, we shall not develop the idea of bus timetables denoting certain traffics. Instead we refer to our previously sketched model of traffics (Sect. 3.4.2, Pages 36–39).

Route (designations) and bus timetables script potential and actual route travels, respectively script the dispatch of buses and their travelling.

Bus timetables can also be seen as a form of contracts between the bus operators offering the bus services and potential and actual passengers, with the contract promising timely transport. In the next section, Sect. 3.5.4, we shall sketch a language of bus service contracts and bus service actions implied by such contracts.

3.5.4 Licenses and Contracts 115

Definition: License. *A license is a* ***script***[651] *specifically expressing a permission to act; is freedom of action; is a permission granted by competent authority to engage in a business or occupation or in an activity otherwise unlawful; a document, plate, or tag evidencing a license granted; a grant by the holder of a copyright or patent to another of any of the rights embodied in the copyright or patent short of an assignment of all rights.*

116 Licenses appear more to have morally than legally binding poser.

Definition: Contract. *A contract is a* ***script***[651] *specifically expressing a legally binding agreement between two or more parties — hence a document describing the conditions of the contract; a contract is business arrangement for the supply of goods or services at fixed*
117 *prices, times and locations. In software development a contract specifies what is to be developed: (1) a* ***domain description***[243]*, (2) a* ***requirements prescription***[615]*, or (3) a* ***software***
118 ***design***[688]*; or a combination of these (1–2, 2–3, 1–3). A contract further specifies how it might, or must be developed; criteria for acceptance of what has been developed; delivery dates for the developed items; who the "parties" to the contract are: the* ***client***[116] *and the*
119 ***developer***[227]*, etc.*

For a comprehensive treatment of licenses and contracts we refer to [48, Chapter 10, Sect. 10.6 (Pages 309–326) [84]].

We shall illustrate fragments of a language for bus service contracts.

The background for the bus contract language is the following. In many large cities around Europe the city or provincial government secures public transport in the form of bus services operated by many different private companies. Section 3.5.2 illustrated the concept of bus (service) timetables. The bus services implied by such a timetable, for a city area — with surrounding suburbs etc. — need not be implemented by just one company, but can be contracted, by the city government public transport office, to several companies,
120 each taking care of a subset of the timetable. Different bus operators then take care of non-overlapping parts and all take care of the full timetable. It may even be that extra buses need be scheduled, on the fly, in connection with major sports or concert or other events. Bus operators may experience vehicle breakdowns or bus driver shortages and may be forced to subcontract other, even otherwise competing bus operators to "step in" and alleviate the problem.

Contracts

121 Schematically we may represent a bus contract as follows:

Contract cn **between contractee** ci **and contractor** cj:
This contract contracts cj **in the period** [t,t′] **to**
perform the following services with respect to timetable tt:

operate bus lines $\{blj_1,blj_2,...,blj_n\}$
subject to the following occasional exceptions:
cancellation of bus tours:
$\{(blj_a,\{bno_{a_1},...,bno_{a_m}\}),...\}$ **subject to conditions** cbt
insertion of bus tours on lines
$\{blj_\alpha,blj_\beta,...,blj_\gamma\}$ **subject to conditions** ibt
subcontracting bus tours on lines
$\{blj_\delta,blj_\phi,...,blj_\omega\}$ **subject to conditions** scbt.

122

111. A bus contract has a header with the distinct names of a contractee and a contractor and a time interval.

112. A bus contract presents a timetable.

113. A bus contract presents a set of bus lines (by their identifiers) such that these are in the timetable.

114. And a bus contract may list one or more of three kinds of "exceptions":

    a) cancellation of one or more named bus tours on one or more bus lines subject to certain (specified) conditions;

    b) insertion of one or more extra bus tours on one or more bus lines subject to certain (specified) conditions;

    c) subcontracting one or more unspecified bus tours on one or more bus lines subject to certain (specified) conditions — to further unspecified contractors.

123

    We abstract the above quoted "one or more of three kinds of exceptions" as one possibly empty clause for each of these alternatives.

115. A bus contract now contains a header, a timetable, the subject bus lines and the exceptions,

116. such that

    a) line names mentioned in the contract are those of the bus lines of the timetable, and

    b) bus (tour) numbers are those of the appropriate bus lines in the timetable.

117. The calendar period is for at least one full day, midnight to midnight.

118. A named contract is a pair of a contract name and a contract.

124

type
111. CNm, CId, D, T, CON
111. CH = CId × CId × (D×D)
112. CT = TT
113. CLs = BLNm-**set**
114. CE = (CA × IN × SC) × CON
114a. CA = BLNm $\overrightarrow{m}$ BNo-**set**
114b. IN = BLNm $\overrightarrow{m}$ BNo-**set**
114c. SC = BLNm-**set**
115. CO′ = CH × CT × CLs × CE
116. CO = {|co:CO′•wf_CO(co)|}
118. NCO = CNm × CO

125

value
116. wf_CO: CO′ → **Bool**
116. wf_CO((ce,cr,(d,d′)),(nd,tbl),cls,((blns,blns′,bls),con)) ≡
113. ce ≠ cr ∧
116a. cls ⊆ **dom** tbl ∧
116b. ∀ bli,bli′:BLNm • bli ∈ **dom** blns ∧ bli′ ∈ **dom** blns′ ⇒
116a. {bli,bli′} ⊆ **dom** tbl ∧
116b. blns(bli) ∪ blns′(bli′) ⊆ **dom** sel_btbtl(tbl(bli)) ∧
116a. bls ⊂ **dom** tbl ∧
117. d < d′

Contractual Actions

126 An bus operator can now perform a number of actions according to a contract. We schematise these:

For contract cn **commence bus tour, line:** bli **and bus no.:** bno

For contract cn **cancel bus tour, line:** bli **and bus no.:** bno

For contract cn **insert extra bus tour, line:** bli **and bus no.:** bno

Subcontract with respect to contract cn **the following:**
Contract cn′: **for the calendar period** [d,d′] **contractee** ci **contracts contractor** cj
to perform the following services with respect to timetable tt:
operate bus lines {blj_1,blj_2,...,blj_n}
subject to the following occasional exceptions:
cancellation of bus tours:
{(blj_c,{bno_{c_1},...,bno_{c_m}}),...} **subject to conditions** cbt
insertion of bus tours on lines

$\{(\text{blj}_i,\{\text{bno}_{i_1},...,\text{bno}_{i_n}\}),...\}$ **subject to conditions** ibt
subcontracting bus tours on lines
$\{\text{blj}_\delta,\text{blj}_\phi,...,\text{blj}_\omega\}$ **subject to conditions** scbt.

127

119. A bus operator action is either a commence, a cancellation, an insertion or a subcontracting action. All actions refer to the (name of) the contract with respect to which the action is contracted.
    a) A commence action designator states the bus line concerned and the bus number of that line.
    b) A cancellation action designator states the bus line concerned and the bus number of that line.
    c) An insertion action designator states the bus line concerned and the bus number of that line — for which an extra bus is to be inserted.[10]
    d) A subcontracting action designator, besides the name of the contract with respect to which the subcontract is a subcontract, state a named contract (whose contract name is unique).

128

type
119. Act = Com | Can | Ins | Sub
119a. Com == mkCom(sel_cn:CNm,sel_bli:BLNm,sel_bno:BNo)
119b. Can == mkCan(sel_cn:CNm,sel_bli:BLNm,sel_bno:BNo)
119c. Ins == mkIns(sel_cn:CNm,sel_bli:BLNm,sel_bno:BNo)
119d. Sub == mkSub(sel_cn:CNm,sel_con:NCO)

Wellformedness of Contractual Actions

129

120. In order to express wellformedness conditions, that is, pre-conditions, for the action designators we introduce a context which map contract names to contracts.

121. Wellformedness of a contract is now expressed with respect to a context.

type
120. CTX = CNm $\xrightarrow[m]{}$ CO
value
121. wf_Act: Act → CTX → **Bool**

130

[10] The insertion of buses in connection with either unscheduled or extraordinary (sports, concerts, etc.) events can be handled by special, initial contracts.

- Let a defined cnm entry in ctx be a contract: ((ce,cr),(nd,tbl),cls,(blns,bls,bls′),(d,d′)).

122. If cmd is a commence command mkCom(cnm,bln,bno), then
   a) contract name cnm must be defined in context ctx;
   b) bus line name bln must be defined in the contract, that is, in cls, and
   c) bus number bno must be defined in the bus table part of table tbl.

122. wf_Act(mkCom(cnm,bln,bno))(ctx) ≡
122a. cnm ∈ **dom** ctx ∧
122. **let** ((ce,cr),(nd,tbl),cls,(blns,bls,bls′),(d,d′)) = ctx(cnm) **in**
122b. bln ∈ cls ∧
122c. bno ∈ **dom** sel_btbl(tbl(bln)) **end**

123. cancellation and insertion commands have the same static wellformedness conditions as have commence command.

123. wf_Act(mkCan(cnm,bln,bno))(ctx) ≡ wf_Act(mkCom(cnm,bln,bno))(ctx)
123. wf_Act(mkIns(cnm,bln,bno))(ctx) ≡ wf_Act(mkCom(cnm,bln,bno))(ctx)

124. If cmd is a subcontract command then

   Let the subcontract command and the cnm named contract in ctx be
   mkSub(cnm,nco:(cnm′,(ce′,cr′,(d″,d‴)),(nd′,tbl′),cls′,(blns′,bls″,bls‴)))
   respectively ((ce,cr,(d,d′)), (nd,tbl), cls, (blns,bls,bls′)).

   a) contract name cnm must be defined in context ctx;
   b) contract name cnm′ must not be defined in context ctx;
   c) the calendar period of the subcontract must be within that of the contract from which it derives;
   d) the net descriptors nd and nd′ must be identical;
   e) the tables tbl and tbl′ and must be identical and
   f) the set, cls′, of bus line names that are the scope of the subcontracting must be a subset of bls′.

124. wf_Act(mkSub(cnm,nco:(cnm′,co:((ce′,cr′,(d″,d‴)),(nd′,tbl′),cls′,(blns′,blns″,bls‴))))))(ctx)
124a. cnm ∈ **dom** ctx ∧
124. **let** co′ = ((ce,cr,(d,d′)),(nd,tbl),cls,(blns,blns′,bls′)) = ctx(cnm) **in**
124b. cnm′ ∉ **dom** tbl ∧
124c. d ≤ d″ ≤ d‴ ≤ d′ ∧
124d. nd′ = nd ∧
124e. tbl′ = tbl ∧
124f. cls′ ⊆ bls′ **end**

134 Wellformedness of contracts, wf_CO(co) and wf_CO(co′), secures other constraints.

We do not here bring any narrated or formalised description of the semantics of contracts and actions. First such a description would be rather lengthy. Secondly a specification would be more of a requirements prescription.

3.6 Management and Organisation 135

Definition: Management. *Management is about resource[620]s: their acquisition[11], scheduling[646] (over time), allocation[33] (over locations), deployment (in performing actions) and disposal ("retirement"). We distinguish between board-directed, strategic, tactical and operational actions. Board-directed actions target mainly financial resources: obtaining new* 136 *funds through conversion of goodwill into financial resources, acquiring and selling "competing" or "supplementary" business units. Strategic actions (see Item 716 on page 220) convert financial resources into production, service supplies and resources and vice-versa — and in this these actions schedule availability of such resources. Tactical actions (see Item 741 on page 222) mainly allocate resources. Operational actions order, monitor and control the deployment of resources in the performance of actions.* 137

Definition: Organisation. *Organisation is about the "grand scale", executive and strategic national, continental or global (world wide) (i) allocation of major resource (e.g., business) units, whether in a hierarchical, in a matrix, or in some other organigram-specified structure, (ii) as well as the clearly defined relations (which information, decisions and actions are transferred) between these units, and (iii) organisational dynamics.* 138

Definition: Management & Organisation. *The composite term management and organisation applies in connection with management[444] as outlined just above and with organisation[500] also outlined above. The term then emphasises the relations between the organisation and management of an enterprise.*

• • •

The borderlines within management actions and across organisation "layouts" are fuzzy.

3.6.1 Transport System Examples 139

We shall only present sketchy examples of management and organsation.[11]

[11] Two remarks: (1) From an albeit superficial study of curricula of a number of business schools it seems, to this author, that the decomposition in *management and organisation* and into *executive, strategic, tactical and operational actions* is not quite the way the *financial, market, sales, product and production* (business administration) aspects of enterprises are looked upon in these schools. (2) We have, in [30], studied issues of management and organisation, and we shall elsewhere study these from the point of view of the signatures of $\mathcal{E}$xecutive, $\mathcal{S}$trategic, $\mathcal{T}$actical and $\mathcal{O}$perational functions as they apply to and results in one or more of the resource types: $\mathbb{F}$inance, $\mathbb{R}$esource, spatial $\mathbb{L}$ocation and $\mathbb{T}$emporal notions of "business environments" ($\rho : ENV$ which binds resource names to $\mathbb{SCHED}$ules) and "business states" ($\sigma : \Sigma$ which binds resource names to resource values) — and where $\mathbb{SCHED}$ules binds resource names to time intervals and [al]locations.

- Executive actions: Deciding on major re-organisation of a transport net (for example introduction of toll roads or freeways, road pricing, major bridges across wide waters [potentially connecting two hitherto unconnected nets], and their management) are executive actions. So are decisions on merging or splitting transport from or into several transport services. Reorganising an enterprise from one characterised by a "deep" hierarchy of management layers (a hierarchy which may very well exemplify highly centralised both administrative and functional monitoring and control) into a matrix of two "shallow" hierarchies, one which addresses tactical and operational management and one which addresses executive and strategic management — with the former (the operations) being replicated across geographical areas while the latter applies "globally" — such reorganisations reflect executive actions (but are carried out by strategic and tactical management).

- Strategic actions: Adding or removing transport links, or major reorganisation of bus timetables are strategic actions. Splitting a(n own) contract into what is still to be operated and subcontracting other parts, for definite, to other bus operators are also strategic actions.

- Tactical actions: Insertion and cancellation of bus services are tactical actions. Subcontracting some parts of a timetable demanded service, for a short while, to other bus operators could be considered tactical actions.

- Operational actions: Commencing and thus, in general, allocating drivers to and sending these off on bus services are operational actions. So are announcing insertion of new (unscheduled) and cancellation of scheduled routes.

3.7 Human Behaviour

Definition: Human Behaviour. *By human behaviour we shall here understand the way a human follows the enterprise* ***rules and regulations***[640] *as well as interacts with a* ***machine***[436]*:* ***dutifully*** *honouring specified (machine* ***dialogue***[230] *or)* ***protocol***[561]*s, or* ***negligently*** *so, or* ***sloppily*** *not quite so, or even* ***criminally*** *not so! Human behaviour is a* ***facet***[285] *of the* ***domain***[239]*. We shall thus model human behaviour also in terms of it failing to react properly, i.e., humans as* ***non-deterministic agent***[24]*s!*

3.8 Towards Theories of Domain Facets

3.8.1 A Theory of Intrinsics

3.8.2 Theories of Support Technologies

An Example Traffic (tf:TF), intrinsically, is a total function over some time interval, from time (t:T) to continuously positioned (p:P) vehicles (tn:TN).

Conventional optical sensors sample, at regular intervals, the intrinsic train traffic. The result is a sampled traffic (stf:sTF). Hence the collection of all optical sensors, for any given net, is a partial function from intrinsic (itf) to sampled train traffics (stf).

We need to express quality criteria that any optical sensor technology should satisfy — relative to a necessary and sufficient description of a closeness predicate. 147

> For all intrinsic traffics, itf, and for all optical sensor technologies, og, the following must hold: Let stf be the traffic sampled by the optical gates. For all time points, t, in the sampled traffic, those time points must also be in the intrinsic traffic, and, for all trains, tn, in the intrinsic traffic at that time, the train must be observed by the optical gates, and the actual position of the train and the sampled position must somehow be checkable to be close, or identical to one another.

Since hubs change state with time, n:N, the net needs to be part of any model of traffic. 148

type
  T, TN
  P = HP | LP
  NetTraffic :: net:N × trf:(V $\overrightarrow{m}$ P)
  iTF = T → NetTraffic
  sTF = T $\overrightarrow{m}$ NetTraffic
  oG = iTF $\xrightarrow{\sim}$ sTF
value
  [close] c: NetTraffic × TN × NetTraffic $\xrightarrow{\sim}$ **Bool**
axiom
  ∀ itt:iTF, og:OG • **let** stt = og(itt) **in**
    ∀ t:T • t ∈ **dom** stt •
      t ∈ 𝔻𝕆𝕄 itt ∧ ∀ Tn:TN • tn ∈ **dom** trf(itt(t))
        ⇒ tn ∈ **dom** trf(stt(t)) ∧ c(itt(t),tn,stt(t)) **end**

𝔻𝕆𝕄 is not an RSL operator. It is a mathematical way of expressing the definition set of a general function. Hence it is not a computable function.

Checkability is an issue of testing the optical sensors when delivered for conformance to the closeness predicate, i.e., to the axiom.

General

149 The formal requirements can be narrated:Let Θ_i and Θ_a designate the spaces of intrinsic and actual-world configurations (contexts and states). For each intrinsic configuration model — that we know is support technology assisted — there exists a support technology solution, that is, a total function from all intrinsic configurations to corresponding actual configurations. If we are not convinced that there is such a function then there is little hope that we can trust this technology

type
  Θ_i, Θ_a
  ST = $\Theta_i \rightarrow \Theta_a$
axiom
  $\forall$ sts:ST-**set**, st:ST • st $\in$ sts $\Rightarrow$ $\forall$ θ_i:Θ_i, $\exists$ θ_a:Θ_a • st(θ_i) = θ_a

3.8.3 A Theory of Rules & Regulations 150

There are, abstractly speaking, usually three kinds of languages involved wrt. (i.e., when expressing) rules and regulations (respectively when invoking actions that are subject to rules and regulations). Two languages, Rules and Reg, exist for describing rules, respectively regulations; and one, Stimulus, exists for describing the form of the [always current] 151 domain action stimuli.

A syntactic stimulus, sy_sti, denotes a function, se_sti:STI: $\Theta \rightarrow \Theta$, from any configuration to a next configuration, where configurations are those of the system being subjected to stimulations. A syntactic rule, sy_rul:Rule, stands for, i.e., has as its semantics, its meaning, rul:RUL, a predicate over current and next configurations, $(\Theta \times \Theta) \rightarrow$ **Bool**, where these next configurations have been brought about, i.e., caused, by the stimuli. These stimuli 152 express: If the predicate holds then the stimulus will result in a valid next configuration.

type
  Stimulus, Rule, Θ
  STI = $\Theta \rightarrow \Theta$
  RUL = $(\Theta \times \Theta) \rightarrow$ **Bool**
value
  meaning: Stimulus $\rightarrow$ STI
  meaning: Rule $\rightarrow$ RUL

  valid: Stimulus $\times$ Rule $\rightarrow \Theta \rightarrow$ **Bool**
  valid(sy_sti,sy_rul)(θ) $\equiv$ meaning(sy_rul)(θ,(meaning(sy_sti))(θ))

  valid: Stimulus $\times$ RUL $\rightarrow \Theta \rightarrow$ **Bool**
  valid(sy_sti,se_rul)(θ) $\equiv$ se_rul(θ,(meaning(sy_sti))(θ))

153

A syntactic regulation, sy_reg:Reg (related to a specific rule), stands for, i.e., has as its semantics, its meaning, a semantic regulation, se_reg:REG, which is a pair. This pair consists of a predicate, pre_reg:Pre_REG, where Pre_REG = $(\Theta \times \Theta) \rightarrow$ **Bool**, and a domain configuration-changing function, act_reg:Act_REG, where Act_REG = $\Theta \rightarrow \Theta$, that is, both 154 involving current and next domain configurations. The two kinds of functions express: If the predicate holds, then the action can be applied.

The predicate is almost the inverse of the rules functions. The action function serves 155 to undo the stimulus function.

```
type
  Reg
  Rul_and_Reg = Rule × Reg
  REG = Pre_REG × Act_REG
  Pre_REG = Θ × Θ → Bool
  Act_REG = Θ → Θ
value
  interpret: Reg → REG
```

156 The idea is now the following: Any action of the system, i.e., the application of any stimulus, may be an action in accordance with the rules, or it may not. Rules therefore express whether stimuli are valid or not in the current configuration. And regulations therefore express whether they should be applied, and, if so, with what effort. 157

More specifically, there is usually, in any current system configuration, given a set of pairs of rules and regulations. Let (sy_rul,sy_reg) be any such pair. Let sy_sti be any possible stimulus. And let θ be the current configuration. Let the stimulus, sy_sti, applied in that configuration result in a next configuration, θ', where $\theta' = (\mathsf{meaning}(\mathsf{sy\_sti}))(\theta)$. Let θ' violate the rule, $\sim\mathsf{valid}(\mathsf{sy\_sti},\mathsf{sy\_rul})(\theta)$, then if predicate part, pre_reg, of the meaning of the regulation, sy_reg, holds in that violating next configuration, $\mathsf{pre\_reg}(\theta,(\mathsf{meaning}(\mathsf{sy\_sti}))(\theta))$, then the action part, act_reg, of the meaning of the regulation, sy_reg, must be applied, $\mathsf{act\_reg}(\theta)$, to remedy the situation. 158

```
axiom
  ∀ (sy_rul,sy_reg):Rul_and_Regs •
    let se_rul = meaning(sy_rul),
        (pre_reg,act_reg) = meaning(sy_reg) in
    ∀ sy_sti:Stimulus, θ:Θ •
      ∼valid(sy_sti,se_rul)(θ)
        ⇒ pre_reg(θ,(meaning(sy_sti))(θ))
          ⇒ ∃ nθ:Θ • act_reg(θ)=nθ ∧ se_rul(θ,nθ)
    end
```

159 It may be that the regulation predicate fails to detect applicability of regulations actions. That is, the interpretation of a rule differs, in that respect, from the interpretation of a regulation. Such is life in the domain, i.e., in actual reality

3.8.4 A Theory of Management & Organisation 160

3.8.5 A Theory of Human Behaviour 161

Commensurate with the above, humans interpret rules and regulations differently, and not always "consistently" — in the sense of repeatedly applying the same interpretations.

Our final specification pattern is therefore: 162

type
  Action = Θ $\xrightarrow{\sim}$ Θ-**infset**
value
  hum_int: Rule → Θ → RUL-**infset**
  action: Stimulus → Θ → Θ
  hum_beha: Stimulus × Rules → Action → Θ $\xrightarrow{\sim}$ Θ-**infset**
  hum_beha(sy_sti,sy_rul)(α)(θ) **as** θset
    post
      θset = α(θ) ∧ action(sy_sti)(θ) ∈ θset
      ∧ ∀ θ′:Θ•θ′ ∈ θset ⇒
        ∃ se_rul:RUL•se_rul ∈ hum_int(sy_rul)(θ)⇒se_rul(θ,θ′)

163

The above is, necessarily, sketchy: There is a possibly infinite variety of ways of interpreting some rules. A human, in carrying out an action, interprets applicable rules and chooses one which that person believes suits some (professional, sloppy, delinquent or criminal) intent. "Suits" means that it satisfies the intent, i.e., yields **true** on the pre/post-configuration pair, when the action is performed — whether as intended by the ones who issued the rules and regulations or not. We do not cover the case of whether an appropriate regulation is applied or not