2. An Ontology of Specification Entities

Definition: Ontology.

- In philosophy: A systematic account of Existence.
- To us:
  - An explicit formal specification of how to represent the phenomena and concepts
  - that are assumed to exist in some area of interest (some universe of discourse)
  - and the relationships that hold among them.

Further clarification:
- An ontology is a catalogue of concepts and their relationships
- including properties as relationships to other concepts.

Definition: Specification.

- We use the term ‘specification’
- to cover the concepts of domain descriptions, requirements prescriptions and software designs.
- More specifically a specification is a definition, usually consisting of many definitions.

Definition: Entity. By an entity we shall understand

- either a simple entity,
- an action,
- an event
- or a behaviour.

2.1. Simple Entities

Definition: Simple Entity. By a simple entity we shall loosely understand

- an individual, static or inert dynamic and that simple entities “roughly correspond” to what we shall think of as values.
- We shall further allow simple entities to be
  - either atomic
  - or composite, i.e., in the latter case having decomposable sub-entities.
2. An Ontology of Specification Entities

2.1. Simple Entities

- Simple entities have attributes.
- Composite entities have
  - attributes,
  - sub-entities and
  - a mereology, the latter explains how the sub-entities are formed into the simple entity.

2.1.1. Net, Hubs and Links

1. There are nets, hubs and links.
2. A net contains zero, one or more hubs.
3. A net contains zero, one or more links.

**Type**

1. N, H, L

**Value**

2. obs_Hs: N → H-set
3. obs_Ls: N → L-set

2.1.2. Unique Hub and Link Identifiers

4. There are hub identifiers and there are link identifiers.
5. From a hub one can observe its hub identifier.
6. From a link one can observe its link identifier.
7. Hubs of a net have unique hub identifiers.
8. Links of a net have unique hub identifiers.

**Type**

4. HI, LI

**Value**

5. obs_HI: H → HI
6. obs_LI: L → LI

**Axiom**

9. ∀ n:N, h:H • {h, h'} ⊆ obs_Hs(n) ∧ h ≠ h' ⇒ obs_HI(h) ≠ obs_HI(h')
8. ∀ n:N, l:L • {l, l'} ⊆ obs_Ls(n) ∧ l ≠ l' ⇒ obs_LI(l) ≠ obs_LI(l')

2.1.3. Observability of Hub and Link Identifiers

9. From every hub (of a net) we can observe the identifiers of the zero, one or more distinct links (of that net) that the hub is connected to.

**Value**

9. obs_LSs: H → LI-set
10. From every link (of a net) we can observe the identifiers of the exactly two (distinct) hubs of that net which the link is connected to.

\textbf{value}

10. \( \text{obs}_{\text{HIs}}: L \rightarrow \text{HI-set} \)

\textbf{axiom}

10. \( \forall n:N, l:L \in \text{obs}_{\text{Ls}}(n) \Rightarrow \) \( \text{card} \ \text{obs}_{\text{HIs}}(l) = 2 \) \( \land \forall hi: \text{HI} \in \text{obs}_{\text{HIs}}(l) \Rightarrow \text{H_{exists}}(hi)(n) \)

\textbf{value}

\( \text{H_{exists}}: \text{HI} \rightarrow N \rightarrow \text{Bool} \)

\( \text{H_{exists}}(hi)(n) \equiv \exists h: \text{H} \in \text{obs}_{\text{Hs}}(n) \land \text{obs}_{\text{HI}}(h) = hi \)

\textbf{2.1.5. Hub and Link Attributes}

In preparation for later descriptions, narrative and formal, we make a slight detour to deal with hub and link attributes – but we omit, at present, from describing these attributes.

12. Hub and link attributes, \( \text{HAtrs} \) and \( \text{LAtrs} \), include the hub and link identifiers that can be observed from hubs and links respectively.

13. These can be observed from hubs and links of nets.

14. And these can be provided as arguments when constructing hubs and links.

\textbf{type}

12. \( \text{HAtrs}, \text{LAtrs} \)

\textbf{value}

13. \( \text{obs}_{\text{HAtrs}}: H \rightarrow \text{HAtrs} \)

14. \( \text{obs}_{\text{LAtrs}}: L \rightarrow \text{LAtrs} \)

\textbf{2.1.6. Hub and Link Generators}

15. From \( \{ \) a (full) set of \( \} \) hub attributes

(a) including an empty set of observable link identifiers

one can generate a hub with

(a) the hub identifier being that of the argument hub attributes,

(b) the link identifiers of the hub being argument the empty set of

link identifiers of the hub attributes and

(c) the argument hub attributes being those of the resulting hub.

15. \( \text{genH}: \text{HAtrs} \rightarrow H \)

15. \( \text{genH(hatrs)} \) as \( h \)

15(a). \( \text{pre} \ \text{obs}_{\text{Ls}}(\text{hatrs}) = \{ \} \)

15(a). \( \text{post} \ \text{obs}_{\text{HI}}(h) = \text{obs}_{\text{HI}}(\text{hatrs}) \)

15(b). \( \land \ \text{obs}_{\text{Ls}}(h) = \{ \} \)

15(c). \( \land \ \text{obs}_{\text{HAtrs}}(h) = \text{hatrs} \)
16. From the set of hub attributes and a net one can “similarly” generate a hub which is not a hub of the net.
17. From the set of link attributes one can “similarly” generate a link.
18. From the set of link attributes and a net one can “similarly” generate a link which is not a link of the net.

where the reader is to narrate and formalise the “similarities”!

2.2. States

Definition: State. By a state we shall understand
• a collection of one or more simple entities.

2.3. Actions

Definition: Action. By an action we shall understand
• something which potentially changes a state,
• that is, a function application to a state
• which potentially changes that state.

16. \text{genH}: \text{HAtrs} \rightarrow N \rightarrow H
17. \text{genH}(\text{hatrs})(n) \text{ as } h
18. \text{pre card } \text{obs}_{\text{HI}}(\text{hatrs})=2
19. \text{post } h \not\in \text{obs}_{\text{Hs}}(n)
20. \text{pre card } \text{obs}_{\text{HI}}(\text{hatrs})=2
21. \text{post } l \not\in \text{obs}_{\text{LS}}(n)

2.3.1. Insert Hubs

19. One can insert a hub, $h$, into a net, $n$.

The hub to be inserted
20. must not be a hub of the net and
21. $h$ cannot already be connected to any links.

That is, we can only insert “isolated” hubs.

The result of inserting a hub, $h$, into a net, $n$, is a new net, $n'$.
22. which is like $n$ except that it now also has the hub $h$. 
value
19. insertH: HAtrs → N → N
19. insertH(hatr)(n) as n'
19. let h = genH(hatr)(n) in
20. pre h /∈ obs_Hs(n)
21. ∧ obs_LIs(h) = {}
22. post obs_Ls(n) = obs_Ls(n')
22. ∧ obs_Hs(n') = obs_Hs(n) ∪ {h}
22. ∧ obs_HAtrs(h) = hatr
19. end

Theorem:
• Inserting a proper hub in a well-formed net
• that is, a net satisfying all relevant axioms,
• results in a likewise well-formed net.

value
23. removeH: H → N → N
26. removeH(h)(n) as n'
24. pre h ∈ obs_Hs(n)
25. ∧ obs_LIs(h) = {}
27. post obs_Ls(n) = obs_Ls(n')
28. ∧ obs_Hs(n') = obs_Hs(n) \ {h}

• Please note the almost line-by-line similarity of the insert and remove hub descriptions
• and that the only difference between these descriptions are the
• membership, union, respectively set difference operations (∉, ∈, \).

2.3.2. Remove Hubs
23. One can remove a hub, h, from a net, n.
   The hub to be removed
24. must be a hub of the net and
25. h cannot be connected to any links.
   That is, the hub, h, may earlier – in is membership of the net – have been connected to links, but these must already, at the time of hub removal, have been removed, see below.
   That is, we can only remove “isolated” hubs.
26. The result of removing a hub, h, from a net, n, is a new net, n',
27. which is like n
28. except that it now no longer has hub h.

2.3.3. Insert Links
29. One can insert a link, ℓ, into a net, n.
   The link to be inserted must
30. not be a link of the net,
31. but the observable hub identifiers must be those of hubs of the net.
The result of inserting a link, $\ell$, into a net, $n$, is a new net, $n'$, in which $\ell$ is now a member.

Let $h_{j_i}, h_{k_i}$ be the two (distinct) hub identifiers of $\ell$ and $h_j, h_k$ be the two (distinct) hubs of $n$ which are identified by $h_{j_i}, h_{k_i}$.

All hubs of net $n$ except $h_j, h_k$ are the same as in $n$ and are unchanged in $n'$.

The two hubs $h_j, h_k$ of $n$ become hubs $h'_j, h'_k$ of $n'$ such that only the observable identifiers of connected links have changed to now also include the identifier of link $\ell$, and such that the observed attributes are those of the argument.

value

\[ \text{insertL}: L \times \text{Latrs} \to N \sim N \]

\[ \text{insertL}(l, \text{latrs})(n) \]

\[ \text{pre} \ 1 \notin \text{obs}_Ls(n) \]

\[ \land \text{obs}_Hls(l) \subseteq \text{xtrHls}(n) \]

\[ \text{post} \ \text{obs}_Ls(n') = \text{obs}_Ls(n) \cup \{l\} \]

\[ \land \ \text{let} \ \{h_{j_i}, h_{k_i}\} = \text{obs}_Hls(l) \text{ in} \]

\[ \text{let} \ (h_j, h_k) = (\text{getH}(h_{j_i})(n), \text{getH}(h_{k_i})(n)) \text{ in} \]

\[ \{h_j, h_k\} \subseteq \text{obs}_Hs(n) \]

\[ \land \ \text{obs}_Hs(n) \setminus \{h_j, h_k\} = \text{obs}_Hs(n') \setminus \{h_j, h_k\} \]

\[ \land \ \text{let} \ (h'_j, h'_k) = (\text{getH}(h_{j_i})(n'), \text{getH}(h_{k_i})(n')) \text{ in} \]

\[ \text{obs}_Lls(h_{j_i}) = \text{obs}_Lls(h_{j_i}) \cup \{\text{obs}_L(l)\} \]

\[ \land \ \text{obs}_Lls(h_{k_i}) = \text{obs}_Lls(h_{k_i}) \cup \{\text{obs}_L(l)\} \text{ end end end} \]

\[ \land \ \text{obs}_L\text{Atrs}(l) = \text{latrs} \]

2.3.4. Remove Links

One can remove a link, $\ell$, from a net, $n$.

The link to be removed must be a link of the net.
The result of removing a link, $\ell$, from a net,
42. $n$, is a new net, $n'$,
43. in which $\ell$ is no longer a member.
44. Let $h_j, h_k$ be the two (distinct) hub identifiers of $\ell$ and
45. let $h_j, h_k$ be the two (distinct) hubs of $n$ which are identified by $h_j, h_k$.
46. $h_j, h_k$ are in $n'$.
47. All hubs of net $n$ except $h_j, h_k$ are the same as in $n$ and are unchanged in $n'$.
48. The two hubs $h_j, h_k$ of $n$ become hubs $h_j', h_k'$ of $n'$
49. such that only the observable identifiers of connected links have changed to now no longer include the identifier of link $\ell$.

2.3.5. Two Theorems
2.3.5.1. Idempotency
- With the preconditions satisfied by the insert and remove actions
- one can prove that first inserting a hub (link) into a net and
- then removing that hub (link) from the resulting net restores the original net:

\[
\begin{align*}
\forall \ n,n':N,h:H,l:L \cdot \\
\text{pre} & \quad \text{insertH}(h)(n) \land \text{removeH}(h)(n') \land \text{insertL}(l)(n) \land \text{removeL}(l)(n') \Rightarrow \\
& \quad \text{removeH}(h)(\text{insertH}(h)(n)) = n \land \text{removeL}(l)(\text{insertL}(l)(n))
\end{align*}
\]

2.3.5.2. Reachability
- Any net that satisfies the axioms above
- can be constructed by sequences of insert hub and link actions.

\[
\begin{align*}
\text{theorem} & \quad \forall n,n':N \cdot \text{obs}_L(n,n')=\text{obs}_L(n,n')=\{\} \in \\
& \quad \forall n,N \vdash \text{axioms} \quad 7 \text{ and } 8 \text{ on page } 14; \quad 9 \text{ on page } 15. \quad 10 \text{ on page } 16. \quad \exists h:H^*, l:L^* \cdot \text{let } n' = \text{insertH}(h)(n,n') \text{ in } \text{insertH}(h)(n,n')=n \text{ end end}
\end{align*}
\]

\[
\begin{align*}
\text{insertHs} & \quad H^* \rightarrow N \xrightarrow{\sim} N \\
\text{insertLs} & \quad L^* \rightarrow N \xrightarrow{\sim} N \\
\text{insertHs}(h)(n) & \equiv \text{case } h \text{ of } (l) \rightarrow n, (h') \rightarrow \text{insertHs}(h')(\text{insertH}(h)(n)) \text{ end end}
\end{align*}
\]
Informal proof: An informal proof goes like this:

- Take a net.
- For every hub, \( h \), in that net,
  - let \( h' \) be a version of \( h \) which has
    * the same hub identifier,
    * an empty set of observable link identifiers (of connected links),
    * and otherwise all other attributes of \( h \),
  - let \( h' \) be a member of the list of hubs – and only such hubs.
  - Let every and only such links in \( n \) be members of the list of links.
- Performing first the insertion of all hubs and then the insertions of all links will “turn the trick”!

end of informal proof.

Definition: Event.

- An event is something that occurs instantaneously.
- Events are manifested by certain state changes, and by certain interactions between behaviours or processes.
- The occurrence of events may “trigger” [further] actions.
- How the triggering, i.e., the invocation of functions are brought about is usually left implied, or unspecified.

A mudslide across a railway track or a road segment (i.e., a link) represents an event
- that effectively “removes” the link, or at least a segment of a link.

Similarly if
- a train and/or automobile bridge collapses or
- a tunnel gets flooded or catches fire.

How are we to model such, and other events?

We choose to model the event” “disappearance” of a segment of a link identified by \( l_i:LI \) as the composition of the following actions:

(a) the removal of link \( l:L \) being affected, where \( l_i:LI \) identifies the link in the network;
(b) the insertion of two hubs, \( h',h'':H \), corresponding to “points” (on link \( l:L \)) on either side of the mudslide or bridge – or other; and
(c) the insertion of two links, \( l',l'':L \), between the hubs of the original link and the new hubs.
(d) \( l_i:LI \) must identify a link \( l:L \) of net \( n:N \).

50(b). newH: N → H-set → H
50(b). newH(n)(hs) ≡ let h:H. h \( \not\in \) hs ∧ obs.Lhs(h)=\{\} in h end
50(c). newL: N → L-set → (HI×HI) → L
50(c). newL(n)(ls)(hi',hi'') ≡ let l:L. l \( \not\in \) ls ∧ obs.Hls(l)={hi',hi''} in l end
2.5. Behaviours

Definition: Behaviour.

- By behaviour we shall understand the way in which something functions or operates.
- In the context of domain engineering behaviour is a concept associated with phenomena, in particular manifest entities.
- And then behaviour is that which can be observed about the value of the entity and its interaction with an environment.
- A simple, sequential behaviour is a sequence of zero, one or more actions and events.

2.5.1. Behaviour Prescriptions

- Usually behaviours follow a prescription.
- In the case of net construction we refer to the prescription as a construction plan.

2.5.1.1. Construction Plans

51. The plan for constructing a net can be abstracted as

(a) a map, PLAN, which to each hub identifier associates
(b) a link-to-hub identifier map, LHIM, from the identifiers of links emanating from the hub to identifiers of connected hubs.

51(a). PLAN = HI ℓ LHIM
51(b). LHIM = LI ℓ HI

2.5.1.2. Wellformedness of Construction Plans

52. Wellformed net construction plans satisfy three conditions:

(a) All Links are Two-way Links:
   i. Let \( h_k \) be any hub identifier of the construction plan.
   ii. For all link identifiers, \( l_j \), of the LIHM, \( lhim_k \), mapped into by \( h_k \),
   iii. let \( h_\ell \) be the hub identifier mapped into by \( l_j \) in \( lhim_k \),
   iv. then \( l_j \) is in the link-to-hub-identifier map, \( lhim_\ell \), mapped into by \( h_\ell \),
(b) Using Hub Identifier Occurrences are Defined:

i. Let \( lhim \) be any link-to-hub-identifier map of a construction plan.

ii. For every hub identifier, \( h_i \), mapped to by a link identifier, \( l_j \), in \( lhim \)

iii. there exists a hub identifier, \( h_k \), that maps into \( l_j \); and

(c) No Junk:

- To secure consistency between hub and link identifiers of a construction plan we impose:
  - all the defined hub identifiers of a construction plan are in the range of some link to hub identifier map of that plan;
  - and each of the hub identifiers of some link to hub identifier map are defined in the construction plan are in the range of some link to hub identifier map of that plan.

\[
\begin{align*}
52(a). \quad & \text{all_links_are_two_way_links: PLAN} \rightarrow \text{Bool} \\
52(a). \quad & \text{all_links_are_two_way_links(plan)} \equiv \\
52(a)i. \quad & \forall \, \text{hk:HI} \cdot \text{hk} \in \text{dom plan} \Rightarrow \\
52(a)ii. \quad & \forall \, \text{lj:LI} \cdot \text{lj} \in \text{dom plan(hk)} \Rightarrow \\
52(a)iii. \quad & \text{let} \; \text{hl} = (\text{plan(hk)})(\text{lj}) \; \text{in} \\
52(a)iv. \quad & \text{lj} \in \text{dom plan(hl)} \; \text{end} \\
52(b). \quad & \text{hub_identifier_occurrences_are_defined: PLAN} \rightarrow \text{Bool} \\
52(b). \quad & \text{hub_identifier_occurrences_are_defined(plan)} \equiv \\
52(b)i. \quad & \forall \, \text{hlim:HLIM} \cdot \text{hlim} \in \text{rng plan} \\
52(b)ii. \quad & \forall \, \text{lj:LI} \cdot \text{lj} \in \text{dom hlim} \Rightarrow \\
52(b)iii. \quad & \exists \, \text{hk:HI} \cdot \text{hk} \in \text{dom plan} \land \text{lj} \in \text{dom plan(hk)} \\
52(c). \quad & \text{no_junk: PLAN} \rightarrow \text{Bool} \\
52(c). \quad & \text{no_junk(plan)} \equiv \text{dom plan} = \bigcup \{ \text{rng(plan(hi))} | \text{hi:HI} \} \cdot \text{hi} \in \text{dom plan}.
\end{align*}
\]
We therefore augment construction plans to also reveal these attributes.

**type**

\[ \text{APLAN} = \text{PLAN} \times \text{HInfo} \times \text{LInfo} \]

\[ \text{HInfo} = \text{HI} \times \text{HAttrs} \]

\[ \text{LInfo} = \text{LI} \times \text{LAttrs} \]

53. The wellformedness of an augmented plan secures that

(a) all hubs identifiers defined in the construction plan are “detailed” in the hub information component, and that

(b) all links identifiers used in the construction plan are “detailed” in the link information component.

**value**

53. \( \text{wf}_{\text{APLAN}} : \text{APLAN} \rightarrow \text{Bool} \)

53. \( \text{wf}_{\text{APLAN}}(\text{plan}, \text{hinfo}, \text{linfo}) \equiv \)

53(a). \( \text{dom} \text{plan} = \text{dom} \text{hinfo} \land \)

53(b). \( \cup\{\text{dom} \text{lhim} | \text{lhim} : \text{LHIM} \land \text{lhim} \in \text{rang} \text{plan}\} = \text{dom} \text{linfo} \)

55. A net construction behaviour can be (functionally and non-deterministically) modelled as

(a) a sequence of hub insertions followed by

(b) a sequence of link insertions.

**value**

55. \( \text{net}_{\text{construction}} : \text{HInfo} \times \text{LInfo} \rightarrow (\text{HI-set} \times \text{LI-set}) \rightarrow \text{N} \rightarrow \text{N} \)

55. \( \text{net}_{\text{construction}}(\text{hinfo}, \text{linfo})(\text{his}, \text{lis})(n) \equiv \)

55. **case** \( \text{his}, \text{lis} \) of

55(a). \( \{\text{hi}\} \cup \text{his}, \_ \) →

55(a). \( \text{net}_{\text{construction}}(\text{hinfo}, \text{linfo})(\text{his}, \text{lis})(\text{insertH}(\text{hinfo}(\text{hi}))(\text{n})), \)

55(b). \( \{\}, \text{lis} \) →

55(b). \( \text{net}_{\text{construction}}(\text{hinfo}, \text{linfo})(\{\}, \text{lis})(\text{insertL}(\text{linfo}(\text{li}))(\text{n})), \)

55. \( \{\}, \{\} \) → \( n \)

55. **end**
The net_construction function is initialised with the full sets of hub and link identifiers and with an empty net:

\[
\text{net_construction}(\text{hinfo, linfo})(\text{dom hinfo}, \text{dom linfo})(\text{n}_\text{nil})
\]

value
\[
n_\text{nil}: \text{N} \cdot \text{obs_Hs}(n_\text{nil}) = \{\} = \text{obs_Ls}(n_\text{nil})
\]

- The net_construction behaviour shown above defines only a subset of all the valid behaviours that will construct a net according to the augmented plan \((\text{plan, hinfo, linfo})\).
- Other valid behaviours would start with constructing at least two hubs but could then go on to construct some of the (zero, one or more) links that connect some of the already constructed hubs, etcetera.
- We challenge the reader to precisely narrate and formally define such net_construction behaviours.

---

**B Slide Table-of-Contents**

<table>
<thead>
<tr>
<th>Contents</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lect. # 1: COVER &amp; INTRODUCTION</td>
<td>0</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>1.1 The Problem</td>
<td>5</td>
</tr>
<tr>
<td>1.2 The Triptych Approach</td>
<td>6</td>
</tr>
<tr>
<td>Lect. # 2: ONTOLOGY</td>
<td>8</td>
</tr>
<tr>
<td>2 An Ontology of Specification Entities</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Simple Entities</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1 Net, Hubs and Links</td>
<td>13</td>
</tr>
<tr>
<td>2.1.2 Unique Hub and Link Identifiers</td>
<td>14</td>
</tr>
<tr>
<td>2.1.3 Observability of Hub and Link Identifiers</td>
<td>15</td>
</tr>
<tr>
<td>2.1.4 A Theorem</td>
<td>17</td>
</tr>
<tr>
<td>2.1.4.1 Links imply Hubs</td>
<td>17</td>
</tr>
<tr>
<td>2.1.5 Hub and Link Attributes</td>
<td>18</td>
</tr>
<tr>
<td>2.1.6 Hub and Link Generators</td>
<td>19</td>
</tr>
<tr>
<td>2.2 States</td>
<td>22</td>
</tr>
<tr>
<td>2.3 Actions</td>
<td>22</td>
</tr>
<tr>
<td>2.3.1 Insert Hubs</td>
<td>23</td>
</tr>
<tr>
<td>2.3.2 Remove Hubs</td>
<td>25</td>
</tr>
<tr>
<td>2.3.3 Insert Links</td>
<td>27</td>
</tr>
<tr>
<td>2.3.4 Remove Links</td>
<td>31</td>
</tr>
<tr>
<td>2.3.5 Two Theorems</td>
<td>34</td>
</tr>
<tr>
<td>2.3.5.1 Idempotency</td>
<td>34</td>
</tr>
<tr>
<td>2.3.5.2 Reachability</td>
<td>35</td>
</tr>
<tr>
<td>2.4 Events</td>
<td>37</td>
</tr>
<tr>
<td>2.5 Behaviours</td>
<td>41</td>
</tr>
<tr>
<td>2.5.1 Behaviour Prescriptions</td>
<td>42</td>
</tr>
<tr>
<td>2.5.1.1 Construction Plans</td>
<td>42</td>
</tr>
<tr>
<td>Lect. # 3: DOMAINS: Intrinsics – Rules &amp; Regulations</td>
<td>52</td>
</tr>
<tr>
<td>3 An Ontology of Domain Facets</td>
<td>53</td>
</tr>
<tr>
<td>3.1 What Can Be Observed</td>
<td>57</td>
</tr>
<tr>
<td>3.2 Intrinsics</td>
<td>58</td>
</tr>
<tr>
<td>3.2.1 Net Topology Descriptors</td>
<td>59</td>
</tr>
<tr>
<td>3.2.2 Link States and Link State Spaces</td>
<td>62</td>
</tr>
<tr>
<td>3.2.3 Hub States and Hub State Spaces</td>
<td>66</td>
</tr>
<tr>
<td>3.2.4 State and State Space Wellformedness</td>
<td>68</td>
</tr>
<tr>
<td>3.2.5 Concrete Types for Simple Entities</td>
<td>69</td>
</tr>
<tr>
<td>3.2.6 Example Hub Crossings</td>
<td>72</td>
</tr>
<tr>
<td>3.2.7 Actions Continued</td>
<td>74</td>
</tr>
<tr>
<td>3.3 Support Technologies</td>
<td>76</td>
</tr>
<tr>
<td>3.3.1 Traffic Signals</td>
<td>77</td>
</tr>
<tr>
<td>3.3.2 Traffic “Control”</td>
<td>81</td>
</tr>
<tr>
<td>3.4 Rules and Regulations</td>
<td>83</td>
</tr>
<tr>
<td>3.4.1 Vehicles</td>
<td>86</td>
</tr>
<tr>
<td>3.4.2 Traffic</td>
<td>88</td>
</tr>
<tr>
<td>3.4.2.1 Wellformedness of Traffic</td>
<td>88</td>
</tr>
<tr>
<td>3.4.2.1.1 Static Wellformedness</td>
<td>89</td>
</tr>
<tr>
<td>3.4.2.1.2 Dynamic Wellformedness</td>
<td>91</td>
</tr>
<tr>
<td>3.4.3 Traffic Rules (I of II)</td>
<td>96</td>
</tr>
<tr>
<td>3.4.4 Another Traffic Regulator</td>
<td>97</td>
</tr>
<tr>
<td>3.4.5 Traffic Rules (II of II)</td>
<td>98</td>
</tr>
<tr>
<td>Lect. # 4: DOMAINS: Scripts – Human Behaviour</td>
<td>98</td>
</tr>
<tr>
<td>4.5 Scripts</td>
<td>99</td>
</tr>
<tr>
<td>4.5.1 Routes as Scripts</td>
<td>100</td>
</tr>
<tr>
<td>4.5.1.1 Paths</td>
<td>100</td>
</tr>
<tr>
<td>3.5.1.2 Routes</td>
<td>103</td>
</tr>
<tr>
<td>3.5.2 Bus Timetables as Scripts</td>
<td>106</td>
</tr>
</tbody>
</table>
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 Detinations ......... 114
3.5.4 License and Contracts .......... 116
3.5.4.1 Contracts .......... 122
3.5.4.2 Contractual Action ..... 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 Intrinsics ......... 147
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

Lect. # 8: REQUIREMENTS – up to and incl. Determination ......... 164

4 An Ontology of Requirements Constructions ......... 165
4.1 Business Process Re-engineering ......... 166
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 ......... 177
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

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 Encodings ......... 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 Basic Concepts ......... 272
1.1.1.1 Atomic Types ......... 272
1.1.1.2 Composite Types ......... 272
1.1.2 Composite Types ......... 272
1.1.2.1 Concrete Types ......... 272
Example 1: Basic Net Attributes ......... 267
Example 2: Composite Net Type Expressions ......... 270
1.1.2.2 Subtypes ......... 279
Example 6: Net Subtypes ......... 292
Example 6: Net Sorts ......... 292

Lect. # 8: RSL: Values & Operations ......... 291
2.1 Concrete RSL Types: Values and Operations ......... 290
2.1.1 Arithmetic ......... 290
2.1.2 Set Expressions ......... 291
2.1.2.1 Set Expressions over Nets ......... 292
2.1.2.2 Set Expressions over Nets ......... 296
Example 8: Set Expressions ......... 297
2.1.3 Cartesian Expressions ......... 298
2.1.3.1 Cartesian Enumerations ......... 299
Example 9: Cartesian Net Types ......... 300
2.1.4 List Expressions ......... 302
2.1.4.1 List Expressions ......... 302
2.1.4.2 List Expressions ......... 302
Example 10: Routes in Nets ......... 304
2.1.5 Map Expressions ......... 309
2.1.5.1 Map Expressions ......... 309
2.1.5.2 Map Expressions ......... 309
Example 1: Concrete Net Type Construction ......... 311
2.1.6 Set Operations ......... 314
2.1.6.1 Set Operations ......... 314
2.1.6.2 Set Expressions ......... 315
2.1.7 Cartesian Operations ......... 316
2.1.8 Map Substitution ......... 317
2.1.8.1 Map Substitution ......... 317
2.1.8.2 Map Substitution ......... 318
2.1.9 Map Operations ......... 319
2.1.9.1 Map Operations ......... 319

Lect. # 9: RSL: Logic, 1-Calculus, FdL, Spsd ......... 320
2 An RSL Predicate Calculus ......... 312
2.1 Propositional Expressions ......... 312
2.2 Simple Predicate Expressions ......... 312
2.3 Quantifier Expressions ......... 312
Example 12: Predicate over Net Quantities ......... 314
1.4 λ-Calculus + Functions ........................................ 327
1.4.1 The λ-Calculus Syntax ..................................... 327
1.4.2 Free and Bound Variables ................................. 328
1.4.3 Substitution .................................................. 329
1.4.4 α-Renaming and β-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
1.5 Other Application Expressions .............................. 345
1.5.1 Simple let Expressions .................................... 345
1.5.2 Recursive let Expressions ................................. 346
1.5.3 Non-deterministic let Clause .............................. 147
1.5.4 Pattern and “Wild Card” let Expressions ............. 348
1.5.5 Conditionals ............................................... 349
Example 16: Choice Pattern Case Expressions: Insert Links
1.5.6 Operator/Operand Expressions ........................... 350
1.6 Imperative Constructs ........................................ 360
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
Example 17: Modeling Connected Links and Hubs .......... 365
1.7.2 Process Definitions ....................................... 369
Example 18: Communicating Hubs, Links and Vehicles .. 371
1.7.3 Process Composition ...................................... 373
Example 19: Modeling Transport Net ........................ 374
1.7.4 Input/Output Events ...................................... 377
Example 20: Modeling Vehicle Movements .................. 378
1.8 Simple RSL Specifications ................................. 384
Example 21: A Neat Little “System” .......................... 386
B Slide Table-of-Contents

Lecture Notes in Software Engineering