

Start of Lecture 2: ONTOLOGY

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. 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.

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*.

- 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.

2.1.2. 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. $\text{obs_HI}: H \rightarrow HI$
6. $\text{obs_LI}: L \rightarrow LI$

axiom

7. $\forall n:N, h, h':H \cdot \{h, h'\} \subseteq \text{obs_Hs}(n) \wedge h \neq h' \Rightarrow \text{obs_HI}(h) \neq \text{obs_HI}(h')$
8. $\forall n:N, l, l':L \cdot \{l, l'\} \subseteq \text{obs_Ls}(n) \wedge l \neq l' \Rightarrow \text{obs_LI}(l) \neq \text{obs_LI}(l')$

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. $\text{obs_Hs}: N \rightarrow H\text{-set}$
3. $\text{obs_Ls}: N \rightarrow L\text{-set}$

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. $\text{obs_LIs}: H \rightarrow LI\text{-set}$

axiom

9. $\forall n:N, h:H \cdot h \in \text{obs_Hs}(n) \Rightarrow \forall li:LI \cdot li \in \text{obs_LIs}(h) \Rightarrow L\text{-exists}(li)(n)$

value

$L\text{-exists}: LI \rightarrow N \rightarrow \mathbf{Bool}$

$L\text{-exists}(li)(n) \equiv \exists l:L \cdot l \in \text{obs_Ls}(n) \wedge \text{obs_LI}(l) = li$

10. From every link (of a net) we can observe the identifiers of the exactly two (distinct) hubs (of that net) that the link is connected to.

value

10. $\text{obs_HIs}: L \rightarrow \text{HI-set}$

axiom

10. $\forall n:N, l:L \cdot l \in \text{obs_Ls}(n) \Rightarrow$

10. $\text{card } \text{obs_HIs}(l)=2 \wedge \forall hi:\text{HI} \cdot hi \in \text{obs_HIs}(l) \Rightarrow \text{H_exists}(hi)(n)$

value

$\text{H_exists}: \text{HI} \rightarrow N \rightarrow \text{Bool}$

$\text{H_exists}(hi)(n) \equiv \exists h:H \cdot h \in \text{obs_Hs}(n) \wedge \text{obs_HI}(h)=hi$

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, **HAttrs** and **LAttrs**, 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 construction hubs and links.

type

12. $\text{HAttrs}, \text{LAttrs}$

value

13. $\text{obs_HAttrs}: H \rightarrow \text{HAttrs}$

14. $\text{obs_LAttrs}: L \rightarrow \text{LAttrs}$

13. $\text{obs_HI}: \text{HAttrs} \rightarrow \text{HI}$

13. $\text{obs_LIs}: \text{HAttrs} \rightarrow \text{LI-set}$

14. $\text{obs_LI}: \text{LAttrs} \rightarrow \text{LI}$

14. $\text{obs_HIs}: \text{LAttrs} \rightarrow \text{HI-set}$

2.1.4. A Theorem

2.1.4.1. Links implies Hubs

11. It follows from the above that if a net has at least one link then it has at least two hubs.

theorem:

11. $\forall n:N \cdot \text{card } \text{obs_Ls}(n) \geq 1 \Rightarrow \text{card } \text{obs_Hs}(n) \geq 2$

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. $\text{genH}: \text{HAttrs} \rightarrow H$

15. $\text{genH}(\text{hattrs}) \text{ as } h$

15(a). **pre** $\text{obs_LIs}(\text{hattrs})=\{\}$

15(a). **post** $\text{obs_HI}(h)=\text{obs_HI}(\text{hattrs})$

15(b). $\wedge \text{obs_LIs}(h)=\{\}$

15(c). $\wedge \text{obs_HAttrs}(h)=\text{hattrs}$

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. $\text{genH: HAtrs} \rightarrow \text{N} \rightarrow \text{H}$
16. $\text{genH}(\text{hatrs})(n) \text{ as } h$
16. **pre** $\text{obs_LIs}(\text{hatrs}) = \{\}$
16. $\wedge \sim \exists h': \text{H} \cdot h' \in \text{obs_Hs}(n) \wedge \text{obs_HI}(h') = \text{obs_HI}(\text{hatrs})$
16. **post** $h \notin \text{obs_Hs}(n)$
16. $\wedge \text{obs_HI}(h) = \text{obs_HI}(\text{hatrs})$
16. $\wedge \text{obs_LIs}(h) = \{\}$
16. $\wedge \text{obs_HAtrs}(h) = \text{hatrs}$

17. $\text{genL: LAtrs} \rightarrow \text{L}$
17. $\text{genL}(\text{latrs}) \text{ as } l$
17. **pre card** $\text{obs_HIs}(\text{latrs}) = 2$
17. **post** $\text{obs_LI}(l) = \text{obs_LI}(\text{latrs})$
17. $\wedge \text{obs_LI}(l) = \text{obs_LI}(\text{latrs})$
17. $\wedge \text{obs_HIs}(l) = \text{obs_HIs}(\text{latrs})$

18. $\text{genL}(\text{latrs})(n) \text{ as } l$
18. **pre card** $\text{obs_LIs}(\text{latrs}) = 2$
18. $\wedge \text{obs_LIs}(\text{latrs}) \subseteq \text{extr_LIs}(n)$
18. **post** $l \notin \text{obs_Ls}(n)$
18. $\wedge \text{obs_LI}(l) = \text{obs_LI}(\text{latrs})$
18. $\wedge \text{obs_HIs}(l) \subseteq \text{obs_HIs}(\text{latrs})$
18. $\wedge \text{obs_LAtrs}(l) = \text{latrs}$

18. $\text{genL: LAtrs} \rightarrow \text{N} \rightarrow \text{L}$

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. $\text{insertH}: \text{HAtrs} \rightarrow \text{N} \xrightarrow{\sim} \text{N}$
19. $\text{insertH}(\text{hatrs})(n)$ **as** n'
19. **let** $h = \text{genH}(\text{hatrs})(n)$ **in**
20. **pre** $h \notin \text{obs_Hs}(n)$
21. $\wedge \text{obs_LIs}(h) = \{\}$
22. **post** $\text{obs_Ls}(n) = \text{obs_Ls}(n')$
22. $\wedge \text{obs_Hs}(n') = \text{obs_Hs}(n) \cup \{h\}$
22. $\wedge \text{obs_HAtrs}(h) = \text{hatrs}$
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. $\text{removeH}: \text{H} \rightarrow \text{N} \xrightarrow{\sim} \text{N}$
26. $\text{removeH}(h)(n)$ **as** n'
24. **pre** $h \in \text{obs_Hs}(n)$
25. $\wedge \text{obs_LIs}(h) = \{\}$
27. **post** $\text{obs_Ls}(n) = \text{obs_Ls}(n')$
28. $\wedge \text{obs_Hs}(n') = \text{obs_Hs}(n) \setminus \{h\}$

- 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 (\notin , \in , \cup respectively \setminus).

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 its 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, ℓ , into a net,

32. n , is a new net, n' ,
33. in which ℓ is now a member.
34. Let h_{j_i}, h_{k_i} be the two (distinct) hub identifiers of ℓ and
35. let h_j, h_k be the two (distinct) hubs of n which are identified by h_{j_i}, h_{k_i} .
36. All hubs of net n except h_j, h_k are the same as in n and are unchanged in n' .
37. The two hubs h_j, h_k of n become hubs h'_j, h'_k of n'
38. such that only the observable identifiers of connected links have changed to now also include the identifier of link ℓ ,
39. and such that the observed attributes are those of the argument.

$\text{xtrHIs}: N \rightarrow \text{HI-set}$

$\text{xtrHIs}(n) \equiv \{\text{obs_HI}(h) | h:H \cdot h \in \text{obs_Hs}(n)\}$

$\text{getH}: \text{HI} \rightarrow N \xrightarrow{\sim} H$

$\text{getH}(hi)(n) \equiv \mathbf{let} \ h:H \cdot h \in \text{obs_Hs}(n) \wedge \text{obs_HI}(h)=hi \ \mathbf{in} \ h \ \mathbf{end}$
 $\mathbf{pre} \ \exists h:H \cdot h \in \text{obs_Hs}(n) \wedge \text{obs_HI}(h)=hi$

value

29. $\text{insertL}: L \times \text{LAtrs} \rightarrow N \xrightarrow{\sim} N$
32. $\text{insertL}(l, \text{latrs})(n) \ \mathbf{as} \ n'$
30. **pre** $l \notin \text{obs_Ls}(n)$
31. $\wedge \text{obs_HIs}(l) \subseteq \text{xtrHIs}(n)$
33. **post** $\text{obs_Ls}(n') = \text{obs_Ls}(n) \cup \{l\}$
34. $\wedge \mathbf{let} \ \{h_{j_i}, h_{k_i}\} = \text{obs_HIs}(l) \ \mathbf{in}$
35. $\mathbf{let} \ (h_j, h_k) = (\text{getH}(h_{j_i})(n), \text{getH}(h_{k_i})(n)) \ \mathbf{in}$
31. $\{h_j, h_k\} \subseteq \text{obs_Hs}(n)$
36. $\wedge \text{obs_Hs}(n) \setminus \{h_j, h_k\} = \text{obs_Hs}(n') \setminus \{h'_j, h'_k\}$
37. $\wedge \mathbf{let} \ (h'_j, h'_k) = (\text{getH}(h_{j_i})(n'), \text{getH}(h_{k_i})(n')) \ \mathbf{in}$
38. $\text{obs_LIs}(h'_k) = \text{obs_LIs}(h_k) \cup \{\text{obs_LI}(l)\}$
38. $\wedge \text{obs_LIs}(h'_j) = \text{obs_LIs}(h_j) \cup \{\text{obs_LI}(l)\} \ \mathbf{end} \ \mathbf{end} \ \mathbf{end}$
39. $\wedge \text{obs_LAtrs}(l) = \text{latrs}$

2.3.4. Remove Links

40. One can remove a link, ℓ , from a net, n .

The link to be removed must

41. be a link of the net.

The result of removing a link, ℓ , from a net,

42. n , is a new net, n' ,
43. in which ℓ is no longer a member.
44. Let h_{j_i}, h_{k_i} be the two (distinct) hub identifiers of ℓ and
45. let h_j, h_k be the two (distinct) hubs of n which are identified by h_{j_i}, h_{k_i} .
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 ℓ .

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:

theorem

$\forall n, n': N, h: H, l: L \cdot$

pre $\text{insertH}(h)(n) \wedge \text{removeH}(h)(n') \wedge \text{insertL}(l)(n) \wedge \text{removeL}(l)(n') \Rightarrow$
 $\text{removeH}(h)(\text{insertH}(h)(n)) = n \wedge \text{removeL}(l)(\text{insertL}(l)(n))$

value

40. $\text{removeL}: L \rightarrow N \xrightarrow{\sim} N$
42. $\text{removeL}(l)(n)$ **as** n'
41. **pre** $l \in \text{obs_Ls}(n)$
43. **post** $\text{obs_Ls}(n') = \text{obs_Ls}(n) \setminus \{l\}$
44. \wedge **let** $\{h_{j_i}, h_{k_i}\} = \text{obs_HIs}(l)$ **in**
45. \quad **let** $(h_j, h_k) = (\text{getH}(h_{j_i})(n), \text{getH}(h_{k_i})(n))$ **in**
46. $\quad \{h_j, h_k\} \subseteq \text{obs_Hs}(n)$
47. $\wedge \text{obs_Hs}(n) \setminus \{h_j, h_k\} = \text{obs_Hs}(n') \setminus \{h_j, h_k\}$
48. \wedge **let** $(h'_j, h'_k) = (\text{getH}(h_{j_i})(n'), \text{getH}(h_{k_i})(n'))$ **in**
49. $\quad \text{obs_LIs}(h'_k) = \text{obs_LIs}(h_k) \setminus \{\text{obs_LI}(l)\}$
49. $\wedge \text{obs_LIs}(h'_j) = \text{obs_LIs}(h_j) \setminus \{\text{obs_LI}(l)\}$ **end end end**

2.3.5.2. Reachability

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

theorem

let $n_{\text{nil}}: N \cdot \text{obs_Hs}(n_{\text{nil}}) = \text{obs_Ls}(n_{\text{nil}}) = \{\}$ **in**
 $\forall n: N \vdash$ **axioms** 7. and 8 on page 14.; 9 on page 15. 10 on page 16. \cdot
 $\exists hl: H^*, ll: L^* \cdot$ **let** $n' = \text{insertHs}(hl)(n_{\text{nil}})$ **in** $\text{insertHs}(ll)(n') = n$ **end**
end

$\text{insertHs}: H^* \rightarrow N \xrightarrow{\sim} N$

$\text{insertLs}: L^* \rightarrow N \xrightarrow{\sim} N$

$\text{insertHs}(hl)(n) \equiv$ **case** hl **of** $\langle \rangle \rightarrow n, \langle h \rangle \wedge hl' \rightarrow \text{insertHs}(hl')(\text{insertH}(h)(n))$ **end**
 $\text{insertLs}(ll)(n) \equiv$ **case** ll **of** $\langle \rangle \rightarrow n, \langle l \rangle \wedge ll' \rightarrow \text{insertLs}(ll')(\text{insertL}(l)(n))$ **end**

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.

- 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?

2.4. Events

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.*

50. 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). $\text{newH}: N \rightarrow \mathbf{H\text{-set}} \rightarrow H$

50(b). $\text{newH}(n)(hs) \equiv \mathbf{let} \ h:H \cdot h \notin hs \wedge \text{obs_LLs}(h)=\{\} \ \mathbf{in} \ h \ \mathbf{end}$

50(c). $\text{newL}: N \rightarrow \mathbf{L\text{-set}} \rightarrow (HI \times HI) \rightarrow L$

50(c). $\text{newL}(n)(ls)(hi',hi'') \equiv \mathbf{let} \ l:L \cdot l \notin ls \wedge \text{obs_HIs}(l)=\{hi',hi''\} \ \mathbf{in} \ l \ \mathbf{end}$

value

50. event_link_disappearance: $LI \rightarrow N \xrightarrow{\sim} N$

50(a). **let** $l = \text{xtrL}(li)(n)$ **in**

50(a). **let** $\{hi', hi''\} = \text{obs_HIs}(l)$ **in**

50(a). **let** $n' = \text{removeL}(l)(n)$ **in**

50(b). **let** $h' = \text{newH}(n)(\text{obs_Hs}(n))$ **in**

50(b). **let** $h'' = \text{newH}(n)(\text{obs_Hs}(n) \cup \{h'\})$ **in**

50(b). **let** $n'' = \text{insertH}(h')(\text{insertH}(h'')(n))$ **in**

50(c). **let** $l' = \text{newL}(n)(\text{obs_Ls}(n))(\text{obs_HI}(h'), hi')$ **in**

50(c). **let** $l'' = \text{newL}(n)(\text{obs_Ls}(n) \cup \{l'\})(\text{obs_HI}(h''), hi'')$ **in**

50(c). **insertL}(l')(insertL}(l'')(n''))** **end end end end end end end end**

50(d). **pre** $li \in \text{xtrLIs}(n)$

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.

type

51(a). $\text{PLAN} = \text{HI} \xrightarrow{\text{map}} \text{LHIM}$

51(b). $\text{LHIM} = \text{LI} \xrightarrow{\text{map}} \text{HI}$

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.

2.5.1.2. 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_ℓ 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_ℓ ,

(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

52(a). $\text{all_links_are_two_way_links}: \text{PLAN} \rightarrow \mathbf{Bool}$ 52(a). $\text{all_links_are_two_way_links}(\text{plan}) \equiv$ 52((a))i. $\forall hk:HI \cdot hk \in \mathbf{dom} \text{ plan} \Rightarrow$ 52((a))ii. $\forall lj:LI \cdot lj \in \mathbf{dom} \text{ plan}(hk) \Rightarrow$ 52((a))iii. **let** $hl = (\text{plan}(hk))(lj)$ **in**52((a))iv. $lj \in \mathbf{dom} \text{ plan}(hl)$ **end**52(b). $\text{hub_identifier_occurrences_are_defined}: \text{PLAN} \rightarrow \mathbf{Bool}$ 52(b). $\text{hub_identifier_occurrences_are_defined}(\text{plan}) \equiv$ 52((b))i. $\forall hlim:HLIM \cdot hlim \in \mathbf{rng} \text{ plan}$ 52((b))ii. $\forall lj:LI \cdot lj \in \mathbf{dom} \text{ lhim} \Rightarrow$ 52((b))iii. $\exists hk:HI \cdot hk \in \mathbf{dom} \text{ plan} \wedge lj \in \mathbf{dom} \text{ plan}(hk)$ 52(c). $\text{no_junk}: \text{PLAN} \rightarrow \mathbf{Bool}$ 52(c). $\text{no_junk}(\text{plan}) \equiv \mathbf{dom} \text{ plan} = \cup \{ \mathbf{rng}(\text{plan}(hi)) \mid hi:HI \cdot hi \in \mathbf{dom} \text{ plan} \}$ (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.

value52. $\text{wf_PLAN}: \text{PLAN} \rightarrow \mathbf{Bool}$ 52. $\text{wf_PLAN}(\text{plan}) \equiv$ 52(a). $\text{all_links_are_two_way_links}(\text{plan}) \wedge$ 52(b). $\text{hub_identifier_occurrences_are_defined}(\text{plan}) \wedge$ 52(c). $\text{no_junk}(\text{plan})$ **2.5.2. Augmented Construction Plans**

- Hubs and links in nets possess attributes (cf. Item 4 on page 14.).
- Some attributes have already been dealt with:
 - the identifiers of hubs and links that can be observed from hubs, respectively links (cf. Items 4. and ?? on page ??.) and
 - the identifiers of hubs that can be observed from links and the identifiers of links that can be observed from hubs (cf. Items 9. and 10 on page 16.).
- In addition hubs and links in nets possess further attributes:
 - spatial location of hubs and links,
 - (locally ascribed) names of hubs and links,
 - lengths of links,
 - etcetera.

We therefore augment construction plans to also reveal these attributes.

type

$$\text{APLAN} = \text{PLAN} \times \text{HInfo} \times \text{LInfo}$$

$$\text{HInfo} = \text{HI} \xrightarrow{\text{map}} \text{HAtrs}$$

$$\text{LInfo} = \text{LI} \xrightarrow{\text{map}} \text{LAtrs}$$

2.5.3. Sequential Construction Behaviours

54. From an augmented construction plan one can “extract” initial information about

- (a) all hubs and
- (b) all links.

value

$$54(a). \text{ xtrH: HI} \rightarrow \text{APLAN} \rightarrow \text{HI} \times \text{HAtrs}, \text{ xtrH}(\text{hi})(_, \text{hinfo}, _) \equiv \text{hinfo}(\text{hi})$$

$$54(b). \text{ xtrL: LI} \rightarrow \text{APLAN} \rightarrow \text{LAtrs}, \text{ xtrL}(\text{li})(_, _, \text{linfo}) \equiv \text{linfo}(\text{li})$$

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 in the link information component.

value

$$53. \text{ wf_APLAN: APLAN} \rightarrow \mathbf{Bool}$$

$$53. \text{ wf_APLAN}(\text{plan}, \text{hinfo}, \text{linfo}) \equiv$$

$$53(a). \quad \mathbf{dom} \text{ plan} = \mathbf{dom} \text{ hinfo} \wedge$$

$$53(b). \quad \cup \{ \mathbf{dom} \text{ lhim} \mid \text{lhim: LHIM} \cdot \text{lhim} \in \text{rang plan} \} = \mathbf{dom} \text{ linfo}$$

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

$$55. \text{ net_construction: HInfo} \times \text{LInfo} \rightarrow (\mathbf{HI\text{-set}} \times \mathbf{LI\text{-set}}) \rightarrow \mathbf{N} \rightarrow \mathbf{N}$$

$$55. \text{ net_construction}(\text{hinfo}, \text{linfo})(\text{his}, \text{lis})(n) \equiv$$

$$55. \quad \mathbf{case} (\text{his}, \text{lis}) \mathbf{of}$$

$$55(a). \quad (\{\text{hi}\} \cup \text{his}', _) \rightarrow$$

$$55(a). \quad \text{net_construction}(\text{hinfo}, \text{linfo})(\text{his}', \text{lis})(\text{insertH}(\text{hinfo}(\text{hi}))(n)),$$

$$55(b). \quad (\{\}, \{\text{li}\} \cup \text{lis}') \rightarrow$$

$$55(b). \quad \text{net_construction}(\text{hinfo}, \text{linfo})(\{\}, \text{lis}')(\text{insertL}(\text{linfo}(\text{li}))(n)),$$

$$55. \quad (\{\}, \{\}) \rightarrow n$$

$$55. \quad \mathbf{end}$$

The `net_construction` function is initialised with the full sets of hub and link identifiers and with an empty net:

$$\text{net_construction}(\text{hinfo}, \text{linfo})(\mathbf{dom} \text{ hinfo}, \mathbf{dom} \text{ linfo})(n_nil)$$

value

$$n_nil:N \cdot \text{obs_Hs}(n_nil) = \{\} = \text{obs_Ls}(n_nil)$$

- 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 (`plan, hinfo, linfo`).
- Other valid behaviours would start with constructing at least two hubs but could then go onto 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

Contents

Lect. # 1: COVER & INTRODUCTION	0
1 Introduction	5
1.1 The Problem	5
1.2 The Triptych Approach	6
Lect. # 2: ONTOLOGY	8
2 An Ontology of Specification Entities	9
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

End of Lecture 2: ONTOLOGY

2.5.1.2 Wellformedness of Construction Plans	43
2.5.2 Augmented Construction Plans	47
2.5.3 Sequential Construction Behaviours	50
Lect. # 3: DOMAINS: Intrinsic – Rules & Regulations	52
3 An Ontology of Domain Facets	53
3.1 What Can Be Observed	57
3.2 Intrinsic	58
3.2.1 Net Topology Descriptors	59
3.2.2 Link States and Link State Spaces	62
3.2.3 Hub States and Hub State Spaces	66
3.2.4 State and State Space Wellformedness	68
3.2.5 Concrete Types for Simple Entities	69
3.2.6 Example Hub Crossings	72
3.2.7 Actions Continued	74
3.3 Support Technologies	76
3.3.1 Traffic Signals	77
3.3.2 Traffic “Control”	81
3.4 Rules and Regulations	83
3.4.1 Vehicles	86
3.4.2 Traffic	88
3.4.2.1 Wellformedness of Traffic	88
3.4.2.1.1 Static Wellformedness	89
3.4.2.1.2 Dynamic Wellformedness	91
3.4.3 Traffic Rules (I of II)	96
3.4.4 Another Traffic Regulator	97
3.4.5 Traffic Rules (II of II)	98
Lect. # 4: DOMAINS: Scripts – Human Behaviour	98
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
Lect. # 5: REQUIREMENTS – up to and incl. Determination	164
4 An Ontology of Requirements Constructions	165
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
Lect. # 6: REQUIREMENTS – from Extension "out"	207
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
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
1.1.2.2 Subtypes	281
Example 5: Net Subtypes	282
1.1.2.3 Sorts — Abstract Types	288
Example 6: Net Sorts	289

Lect. # 8: RSL: Values & Operations	289
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
Lect. # 9: RSL: Logic, λ-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 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: <i>Network Traffic</i>	331
1.4.5 Function Signatures	338
Example 14: <i>Hub and Link Observers</i>	339
1.4.6 Function Definitions	341
Example 15: <i>Axioms over Hubs, Links and Their Observers</i>	344
1.5 Other Applicative Expressions	345
1.5.1 Simple let Expressions	345
1.5.2 Recursive let Expressions	346
1.5.3 Non-deterministic let Clause	347
1.5.4 Pattern and "Wild Card" let Expressions	348
1.5.5 Conditionals	349
Example 16: <i>Choice Pattern Case Expressions: Insert Links</i>	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
Example 17: <i>Modelling Connected Links and Hubs</i>	365
1.7.2 Process Definitions	369
Example 18: <i>Communicating Hubs, Links and Vehicles</i>	371
1.7.3 Process Composition	373
Example 19: <i>Modelling Transport Nets</i>	374
1.7.4 Input/Output Events	377

Example 20: <i>Modelling Vehicle Movements</i>	378
1.8 Simple RSL Specifications	384
Example 21: <i>A Neat Little "System"</i>	388
B Slide Table-of-Contents	397