2 An Ontology of Specification Entities 9

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 concept[152]s and their relationships — including properties as relationships to other concepts.* 10
Definition: Specification. *We use the term 'specification' to cover the concepts of domain description[243]s, requirements prescription[615]s and software design[688]s. More specifically a specification is a definition[210], usually consisting of many definitions.*
Definition: Entity. *By an entity we shall understand either a simple entity[681]*[3]*, an action[12], an event[281] or a behaviour[79].*

2.1 Simple Entities 11

Definition: Simple Entity. *By a simple entity we shall loosely understand an individual, static[708] or inert[367] dynamic[260] and that simple entities "roughly correspond" to what we shall think of as value[802]s. We shall further allow simple entities to be either atomic[63] or composite[133], i.e., in the latter case having decomposable sub-entities. Simple entities have* 12 *attribute[69]s. Composite entities have attribute[69]s, sub-entities and a mereology[451], the latter explains how the sub-entities are formed into the simple entity.*

2.1.1 Net, Hubs and Links 13

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 14

4. There are hub identifiers and there are link identifiers.
5. Hubs of a net have unique hub identifiers.
6. Links of a net have unique hub identifiers.

[3]The superscript [bracketed numbers] refer to Sect. B's Item 681 on page 216.

type
  4. HI, LI
value
  5. obs_HI: H → HI
  6. obs_LI: H → LI
axiom
  5. $\forall$ n:N, h,h′:H • {h,h′}$\subseteq$obs_Hs(n) $\wedge$ h$\neq$h′ $\Rightarrow$ obs_HI(h)$\neq$obs_HI(h′)
  6. $\forall$ n:N, l,l′:L • {l,l′}$\subseteq$obs_Ls(n) $\wedge$ l$\neq$l′ $\Rightarrow$ obs_LI(l)$\neq$obs_LI(l′)

2.1.3 Observability of Hub and Link Identifiers 15

7. 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
  7. obs_LIs: H → LI-**set**
axiom
  7. $\forall$ n:N,h:H•h $\in$ obs_Hs(n) $\Rightarrow$ $\forall$ li:LI•li $\in$ obs_LIs(h) $\Rightarrow$ L_exists(li)(n)
value
  L_exists: LI → N → **Bool**
  L_exists(li)(n) $\equiv$ $\exists$ l:L•l $\in$ obs_Ls(n)$\wedge$obs_LI(l)=li

16

8. 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
  8. obs_HIs: L → HI-**set**
axiom
  8. $\forall$ n:N,l:L•l $\in$ obs_Ls(n) $\Rightarrow$
  8.   **card** obs_HIs(l)=2 $\wedge$ $\forall$ hi:HI•hi $\in$ obs_HIs(l) $\Rightarrow$ H_exists(hi)(n)
value
  H_exists: HI → N → **Bool**
  H_exists(hi)(n) $\equiv$ $\exists$ h:H•h $\in$ obs_Hs(n)$\wedge$obs_HI(h)=hi

2.1.4 A Theorem 17

Links implies Hubs

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

theorem:
  9. $\forall$ n:N • **card** obs_Ls(n)$\geq$1 $\Rightarrow$ **card** obs_Hs(n)$\geq$2

2.1.5 Hub and Link Attributes 18

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.

10. Besides hub and link identifiers we can speak of additional hub and link attributes, HAtrs and LAtrs.
11. These can be observed from hubs and links of nets..
12. And these can be provided as arguments when construction hubs and links.

type
10. HAtrs, LAtrs
value
11. obs_HAtrs: H $\rightarrow$ HAtrs
12. obs_LAtrs: L $\rightarrow$ LAtrs

2.1.6 Hub and Link Generators 19

13. From a hub identifier and a set of hub attributes one can generate a hub.
14. From a hub identifier, a set of hub attributes and a net one can generate a hub which is not a hub of the net.
15. From a link identifier, a pair of known, distinct hub identifiers and a set of link attributes one can generate a link.
16. From a link identifier, a set of hub attributes and a net one can generate a link which is not a link of the net.

20

13. genH: HI $\times$ HAtrs $\rightarrow$ H
13. genH(hi,hatrs) **as** h
13. **post** obs_HI(h)=hi $\wedge$ obs_LIs(h)={} $\wedge$ obs_HAtrs(h)=hatrs

14. genH: HI $\times$ HAtrs $\rightarrow$ N $\rightarrow$ H
14. genH(hi,hatrs)(n) **as** h
14. **pre** h $\notin$ obs_Hs(n)
14. **post** obs_HI(h)=hi $\wedge$ obs_LIs(h)={} $\wedge$ obs_HAtrs(h)=hatrs

15. genL: LI $\times$ (HI$\times$HI) $\times$ LAtrs $\rightarrow$ L
15. genL(li,(hi$'$,hi$''$),latrs) **as** l
15. **pre** hi$'$$\neq$hi$''$ $\wedge$ {hi$'$,hi$''$}$\subseteq$xtrHIs(n)
15. **post** obs_LI(h)=li $\wedge$ obs_HIs(l)={hi$'$,hi$''$} $\wedge$ obs_LAtrs(h)=latrs

```
16. genL: LI × (HI×HI) × LAtrs → N → L
16. genL(li,(hi′,hi″),latrs)(n) as l
16.   pre hi′≠hi″∧ {hi′,hi″}⊆xtrHIs(n)
16.   post obs_LI(l)=li ∧ obs_HIs(l)={hi′,hi″} ∧ obs_LAtrs(h)=latrs ∧ l ∉ obs_Ls(n)
```

2.2 States 21

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**[705]*, that is, a function application to a state which potentially changes that state.*

2.3.1 Insert Hubs 22

17. One can insert a hub, h, into a net, n.

The hub to be inserted

18. must not be a hub of the net and
19. 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',

20. which is like n except that it now also has the hub h.

23

value

```
17. insertH: H × HAtrs → N $\xrightarrow{\sim}$ N
17. insertH(h,hatrs)(n) as n′
18. pre  h ∉ obs_Hs(n) ∧
19.          obs_LIs(h) = {}
20. post obs_Ls(n)=obsLs(n′) ∧
20.          obs_Hs(n′)=obs_Hs(n)∪{h} ∧
20.          obs_HAtrs(h)=hatrs
```

The argument hub h in insertH(h,hatrs)(n) may have been "concocted" from using either genH(hi,hatrs) or genH(hi,hatrs)(n).

2.3.2 Remove Hubs 24

21. One can remove a hub, h, from a net, n.

    The hub to be removed

22. must be a hub of the net and
23. h cannot be connected to any links.

    That is, the hub, h, may earlier – it is membership of the net – have been connected to links, but these must already, at the time or hub removal, have been removed, see below.

    That is, we can only remove "isolated" hubs.

    The result of removing a hub, h, from a net, n, is a new net, n',

24. which is like n except that it now no longer has the h.

25

value

```
21.  removeH: H → N $\overset{\sim}{\rightarrow}$ N
21.  removeH(h)(n) **as** n′
22.  **pre**  h ∈ obs_Hs(n) ∧
23.          obs_LIs(h) = {}
24.  **post** obs_Ls(n)=obsLs(n′) ∧ 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 ($\notin$, $\in$, $\cup$ respectively $\setminus$).

2.3.3 Insert Links 26

25. One can insert a link, ℓ, into a net, n.

The link to be inserted must

26. not be a link of the net,
27. but the observable hub identifiers must be those of hubs of the net.

27

The result of inserting a link, ℓ, into a net,

28. n, is a new net, n',
29. in which ℓ is now a member.
30. Let h_{j_i}, h_{k_i} be the two (distinct) hub identifiers of ℓ and

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

28

value

```
25.  insertL: L × LAtrs → N $\xrightarrow{\sim}$ N
28.  insertL(l,latrs)(n) **as** n′
26.  **pre**  l ∉ obs_Ls(n) ∧
27.        obs_HIs(l)⊆xtrHIs(n)
29.  **post** obs_Ls(n′) = obs_Ls(n) ∪ {l} ∧
30.        **let** {hji,hki}=obs_HIs(l) **in**
31.        **let** (hj,hk) = (getH(hji)(n),getH(hki)(n)) **in**
27.        {hj,hk}⊆obs_Hs(n) ∧
32.        obs_Hs(n)\{hj,hk} = obs_Hs(n′)\{hj,hk} ∧
33.        **let** (hj′,hk′) = (getH(hji)(n′),getH(hki)(n′)) **in**
34.        obs_LIs(hk′) = obs_LIs(hk′) ∪ {obs_LI(l)}
34.        obs_LIs(hj′) = obs_LIs(hj′) ∪ {obs_LI(l)} **end end end**
35.        obs_LAtrs(l) = latrs
```

29

```
xtrHIs: N → HI-**set**
xtrHIs(n) ≡ {obs_HI(h)|h:H•h ∈ obs_Hs(n)}

getH: HI → N $\xrightarrow{\sim}$ H
getH(hi)(n) ≡ **let** h:H • h ∈ obs_Hs(n) ∧ obs_HI(h)=hi **in** h **end**
   **pre** ∃ h:H • h ∈ obs_Hs(n) ∧ obs_HI(h)=hi
```

2.3.4 Remove Links 30

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

The link to be removed must

37. be a link of the net.

31

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

38. n, is a new net, n',
39. in which ℓ is no longer a member.
40. Let h_{j_i}, h_{k_i} be the two (distinct) hub identifiers of ℓ and
41. let h_j, h_k be the two (distinct) hubs of n which are identified by h_{j_i}, h_{k_i}.
42. h_j, h_k are in n'.
43. All hubs of net n except h_j, h_k are the same as in n and are unchanged in n'.
44. The two hubs h_j, h_k of n become hubs h'_j, h'_k of n'
45. such that only the observable identifiers of connected links have changed to now no longer include the identifier of link ℓ.

32

value
```
36.  removeL: L → N $\xrightarrow{\sim}$ N
38.  removeL(l)(n) **as** n′
37.  **pre**  l ∈ obs_Ls(n)
39.  **post** obs_Ls(n′) = obs_Ls(n) \ {l} ∧
40.        **let** {hji,hki}=obs_HIs(l) **in**
41.        **let** (hj,hk) = (getH(hji)(n),getH(hki)(n)) **in**
42.        {hj,hk}⊆obs_Hs(n) ∧
43.        obs_Hs(n)\{hj,hk} = obs_Hs(n′)\{hj,hk} ∧
44.        **let** (hj′,hk′) = (getH(hji)(n′),getH(hki)(n′)) **in**
45.        obs_LIs(hk′) = obs_LIs(hk′) \ {obs_LI(l)}
45.        obs_LIs(hj′) = obs_LIs(hj′) \ {obs_LI(l)} **end end end**
```

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

2.3.5 Two Theorems 33

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
```
∀ n,n′:N,h:H,l:L •
    **pre** insertH(h)(n) ∧ removeH(h)(n′) ∧ insertL(l)(n) ∧ removeL(l)(n′) ⇒
    removeH(h)(insertH(h)(n)) = n ∧ removeL(l)(insertL(l)(n))
```

Reachability 34 Any net that satisfies the axioms above can be constructed by sequences of insert hub and link actions.

```
theorem
    let n_nil:N • obs_Hs(n_nil)=obs_Ls(n_nil)={} in
    ∀ n:N ⊢ axioms 5. and 6 on page 13.; 7 on page 14. 8 on page 14. •
        ∃ hl:H*, ll:L* • let n′ = insertHs(hl)(n_nil) in insertHs(hl)(n′)=n end
    end

    insertHs: H* → N $\xrightarrow{\sim}$ N
    insertLs: L* → N $\xrightarrow{\sim}$ N

    insertHs(hl)(n) ≡ case hl of ⟨⟩ → n, ⟨h⟩^hl′ → insertHs(hl′)(insertH(h)(n)) end
    insertLs(ll)(n) ≡ case ll of ⟨⟩ → n, ⟨l⟩^ll′ → insertLs(ll′)(insertL(l)(n)) end
```

35 **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.**

2.4 Events 36

Definition: Event. *An event is something that occurs instantaneously. Events are manifested by certain state[705] changes, and by certain interaction[392]s between behaviour[79]s or process[544]es. The occurrence of events may "trigger" [further] actions. How the triggering, i.e., the invocation[402] of functions are brought about is usually left implied, or unspecified.* 37

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.

38 How are we to model such, and other events?

46. 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.

```
46b. newH: N → H-set → N
46b. newH(n)(hs) ≡ let h:H • h ∉ hs ∧ obs_LIs(h)={} in h end
46c. newL: N → L-set → (HI×HI) → L
46c. newL(n)(ls)(hi′,hi″) ≡ let l:L • l ∉ ls ∧ obs_HIs(l)={hi′,hi″} in l end
```

39

value

```
46.  event_link_disappearance: LI → N ~→ N
46a.   let l = xtrL(li)(n) in
46a.   let {hi′,hi″} = obs_HIs(l) in
46a.   let n′ = removeL(l)(n) in
46b.   let h′= newH(n)(obs_Hs(n)) in
46b.   let h″ = newH(n)(obs_Hs(n)∪{h′}) in
46b.   let n″ = insertH(h′)(insertH(h″)(n)) in
46c.   let l′ = newL(n)(obs_Ls(n))(obs_HI(h′),hi′) in
46c.   let l″ = newL(n)(obs_Ls(n)∪{l′})(obs_HI(h″),hi″) in
46c.   insertL(l′)(insertL(l″)(n″)) end end end end end end end end
46d. pre li ∈ xtrLIs(n)
```

The newH and newL generator (or constructor) functions are simplified versions of more realistic such functions. Hubs and links, as we shall see, have attributes beyond those obs_HI, obs_LI, obs_LIs and obs_HIs. Proper newH and newL generator definitions must express that initial values be ascribed to these other attributes. Examples of further hub and link attributes are: spatial location, name[4], mode[5], length for links, etcetera. So, eventually, the definitions of the newH and newL constructors will have to be redefined.

There will be very many other kinds of events in connection with transportation.

$\boxed{\text{MORE TO COME}}$

2.5 Behaviours 40

Definition: Behaviour. *By behaviour we shall understand the way in which something functions or operates. In the context of domain engineering[248] behaviour is a concept associated with phenomena, in particular manifest entities[272]. And then behaviour is that which can be observed about the value[802] of the entity[272] and its interaction[392] with an environment[275]. A simple, sequential behaviour is a sequence of zero, one or more actions and events.*

[4]Names of hubs and links must not be confused with hub and link identifiers: Two or more hubs and/or links may have the same name. Hub and link identifiers may be thought of as abstractions of some composition of locations and names in that no two hubs and/or links can "occupy" "overlapping" locations, that is, locations are unique.

[5]whether road, railway, shipping or air traffic hubs and links, or, even combinations of these

2.5.1 Behaviour Prescriptions 41

Usually behaviours follow a prescription.

In the case of net construction we refer to the prescription as a construction plan.

Construction Plans

47. 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
47a. PLAN = HI $\underset{m}{\rightarrow}$ LHIM
47b. LHIM = LI $\underset{m}{\rightarrow}$ HI-**set**

The hub identifiers of the definition set of construction plans are called the defining occurrences of hub identifiers.

The hub identifiers of the ranges of link-to-hub identifier map are called the using occurrences of hub identifiers.

Wellformedness of Construction Plans 42

48. 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,

43

    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

44

    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.

```
value
48.    wf_PLAN: PLAN → Bool
48.    wf_PLAN(plan) ≡
48a.     all_links_are_two_way_links(plan) ∧
48b.     hub_identifier_occurrences_are_defined(plan) ∧
48c.     no_junk(plan)
```

45

```
48a.    all_links_are_two_way_links: PLAN → Bool
48a.    all_links_are_two_way_links(plan) ≡
48(a)i.     ∀ hk:HI • hk ∈ dom plan ⇒
48(a)ii.       ∀ lj:LI • lj ∈ dom plan(hk) ⇒
48(a)iii.         let hl = (plan(hk))(lj) in
48(a)iv.          lj ∈ dom plan(hl) end
```

```
48b.    hub_identifier_occurrences_are_defined: PLAN → Bool
48b.    hub_identifier_occurrences_are_defined(plan) ≡
48(b)i.     ∀ hlim:HLIM•hlim ∈ rng plan
48(b)ii.       ∀ lj:LI • lj ∈ dom lhim ⇒
48(b)iii.         ∃ hk:HI • hk ∈ dom plan ∧ lj ∈ dom plan(hk)
```

```
48c.    no_junk: PLAN → Bool
48c.    no_junk(plan) ≡ dom plan = ∪{rng(plan(hi))|hi:HI•hi ∈ dom plan}
```

2.5.2 Augmented Construction Plans 46

Hubs and links in nets possess attributes (cf. Item 4 on page 13.). 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 5 on page 13.) and the identifiers of hubs that can be observed from links and the identifiers of links that can be observed from hubs (cf. Items 7. and 8 on page 14.).

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.

47

We therefore augment construction plans to also reveal these attributes.

type

```
    APLAN = PLAN × HInfo × LInfo
    HInfo = HI $\overrightarrow{m}$ HAtrs
    LInfo = LI $\overrightarrow{m}$ LAtrs
```

48

49. 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

```
49.   wf_APLAN: APLAN → **Bool**
49.   wf_APLAN(plan,hinfo,linfo) ≡
49a.      **dom** plan = **dom** hinfo ∧
49b.      ∪{**dom** lhim|lhim:LHIM•lhim ∈ rang plan}=**dom** linfo
```

2.5.3 Sequential Construction Behaviours 49

50. From an augmented construction plan one can “extract” initial information about
    a) all hubs and
    b) all links.

value

```
50a. xtrH: HI → APLAN → HI × HAtrs, xtrH(hi)(_,hinfo,_) ≡ hinfo(hi)
50b. xtrL: LI → APLAN → LAtrs, xtrL(li)(_,_,linfo) ≡ linfo(li)
```

50

51. 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

```
51.  net_construction: HInfo×LInfo → (HI-**set**×LI-**set**) → N → N
51.  net_construction(hinfo,linfo)(his,lis)(n) ≡
51.     **case** (his,lis) **of**
51a.        ({hi}∪ his′,_) →
51a.                net_construction(hinfo,linfo)(his′,lis)(insertH′(hi,hinfo(hi))(n)),
51b.      ({},{li}∪ lis′) →
51b.          net_construction(hinfo,linfo)({},lis′)(insertL′(li,linfo(li))(n)),
51.       ({},{}) → n
51.     end
```

51

```
insertH′: HI × HATRS → N → N
insertH′(hi,hatrs)(n) ≡
    insertH(genH(hi,hatrs)(n),hatrs)(n)

insertL′: LI × (HI×HI) × LATRS → N → N
insertL′(li,(hi′,hi″),latrs)(n) ≡
    insertL(genL(li,{hi′,hi″},latrs)(n),latrs)(n)
```

52

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

```
   net_construction(plan,hinfo,linfo)(dom hinfo,dom linfo)(n_nil)
value
   n_nil:N • obs_Hs(n_nil) = {} = 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 precise narrate and formally define such net_construction behaviours.