Start of Lecture 2: ONTOLOGY

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.*

- *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
 7. ∀ n:N, h,h′:H • {h,h′}⊆obs_Hs(n) ∧ h≠h′ ⇒ obs_HI(h)≠obs_HI(h′)
 8. ∀ n:N, l,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_LIs: H → LI-**set**
axiom
 9. ∀ n:N,h:H•h ∈ obs_Hs(n) ⇒ ∀ li:LI•li ∈ obs_LIs(h) ⇒ L_exists(li)(n)
value
 L_exists: LI → N → **Bool**
 L_exists(li)(n) ≡ ∃ l:L•l ∈ obs_Ls(n)∧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. obs_HIs: L → HI-**set**
axiom
10. ∀ n:N,l:L·l ∈ obs_Ls(n) ⇒
10. **card** obs_HIs(l)=2 ∧ ∀ hi:HI·hi ∈ obs_HIs(l) ⇒ H_exists(hi)(n)
value
H_exists: HI → N → **Bool**
H_exists(hi)(n) ≡ ∃ h:H·h ∈ obs_Hs(n)∧obs_HI(h)=hi

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. ∀ n:N · **card** obs_Ls(n)≥1 ⇒ **card** obs_Hs(n)≥2

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, HAtrs and LAtrs, include the hub and link identifiers that can be observed from hubs and links, respecively.
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. HAtrs, LAtrs
value
13. obs_HAtrs: H → HAtrs
14. obs_LAtrs: L → LAtrs
13. obs_HI: HAtrs → HI
13. obs_LIs: HAtrs → LI-**set**
14. obs_LI: LAtrs → LI
14. obs_HIs: LAtrs → HI-**set**

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. genH: HAtrs → H
15. genH(hatrs) **as** h
15(a). **pre** obs_LIs(hatrs)={}
15(a). **post** obs_HI(h)=obs_HI(hatrs)
15(b). ∧ obs_LIs(h)={}
15(c). ∧ obs_HAtrs(h)=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"!

16. genH: HAtrs → N → H
16. genH(hatrs)(n) **as** h
16. **pre** obs_LIs(hatrs)={}
16. ∧ ∼∃ h′:H•h′ ∈ obs_Hs(n) ∧ obs_HI(h′)=obs_HI(hatrs)
16. **post** h ∉ obs_Hs(n)
16. ∧ obs_HI(h)=obs_HI(hatrs)
16. ∧ obs_LIs(h)={}
16. ∧ obs_HAtrs(h)=hatrs

17. genL: LAtrs → L
17. genL(latrs) **as** l
17. **pre** **card** obs_HIs(latrs)=2
17. **post** obs_LI(l)=obs_LI(latrs)
17. ∧ obs_LI(l)=obs_LI(latrs)
17. ∧ obs_HIs(l)=obs_HIs(latrs)

18. genL: LAtrs → N → L
18. genL(latrs)(n) **as** l
18. **pre** **card** obs_LIs(latrs)=2
18. ∧ obs_LIs(latrs)⊆xtr_LIs(n)
18. **post** l ∉ obs_Ls(n)
18. ∧ obs_LI(l)=obs_LI(latrs)
18. ∧ obs_HIs(l)⊆obs_HIs(latrs)
18. ∧ obs_LAtrs(l)=latrs

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.*

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 $\rightarrow$ N $\stackrel{\sim}{\rightarrow}$ N
19. insertH(hatrs)(n) **as** n′
19. **let** h = genH(hatrs)(n) **in**
20. **pre** h $\notin$ obs_Hs(n)
21. $\wedge$ obs_LIs(h) = {}
22. **post** obs_Ls(n)=obsLs(n′)
22. $\wedge$ obs_Hs(n′)=obs_Hs(n)$\cup${h}
22. $\wedge$ obs_HAtrs(h)=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.

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.

value

23. removeH: H $\rightarrow$ N $\stackrel{\sim}{\rightarrow}$ N
26. removeH(h)(n) **as** n′
24. **pre** h $\in$ obs_Hs(n)
25. $\wedge$ obs_LIs(h) = {}
27. **post** obs_Ls(n)=obsLs(n′)
28. $\wedge$ 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 \).

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.

value

```
29.  insertL: L × LAtrs → N ~→ N
32.  insertL(l,latrs)(n) as n′
30.  pre  l ∉ obs_Ls(n)
31.     ∧ obs_HIs(l)⊆xtrHIs(n)
33.  post obs_Ls(n′) = obs_Ls(n) ∪ {l}
34.      ∧ let {hji,hki}=obs_HIs(l) in
35.        let (hj,hk) = (getH(hji)(n),getH(hki)(n)) in
31.        {hj,hk}⊆obs_Hs(n)
36.      ∧ obs_Hs(n)\{hj,hk} = obs_Hs(n′)\{hj,hk}
37.      ∧ let (hj′,hk′) = (getH(hji)(n′),getH(hki)(n′)) in
38.        obs_LIs(hk′) = obs_LIs(hk′) ∪ {obs_LI(l)}
38.      ∧ obs_LIs(hj′) = obs_LIs(hj′) ∪ {obs_LI(l)} end end end
39.      ∧ obs_LAtrs(l) = latrs
```



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

getH: HI → N ~→ 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

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.

value

```
40. removeL: L → N ∼→ N
42. removeL(l)(n) as n′
41. pre  l ∈ obs_Ls(n)
43. post obs_Ls(n′) = obs_Ls(n) \ {l}
44.     ∧ let {hji,hki}=obs_HIs(l) in
45.         let (hj,hk) = (getH(hji)(n),getH(hki)(n)) in
46.         {hj,hk}⊆obs_Hs(n)
47.     ∧ obs_Hs(n)\{hj,hk} = obs_Hs(n′)\{hj,hk}
48.     ∧ let (hj′,hk′) = (getH(hji)(n′),getH(hki)(n′)) in
49.         obs_LIs(hk′) = obs_LIs(hk′) \ {obs_LI(l)}
49.     ∧ obs_LIs(hj′) = obs_LIs(hj′) \ {obs_LI(l)} end end end
```


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

```
∀ 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))
```


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_nil:N • obs_Hs(n_nil)=obs_Ls(n_nil)={} in
∀ n:N ⊢ axioms  7. and 8 on page 14.; 9 on page 15. 10 on page 16. •
  ∃ hl:H*, ll:L* • let n′ = insertHs(hl)(n_nil) in insertHs(ll)(n′)=n end
end

insertHs: H* → N ∼→ N
insertLs: L* → N ∼→ 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
```

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

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?

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

value

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


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.

type

```
51(a).  PLAN = HI  m→  LHIM
51(b).  LHIM = LI  m→  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_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

(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
52. wf_PLAN: PLAN → **Bool**
52. wf_PLAN(plan) ≡
52(a). all_links_are_two_way_links(plan) ∧
52(b). hub_identifier_occurrences_are_defined(plan) ∧
52(c). no_junk(plan)

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

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

52(c). no_junk: PLAN → **Bool**
52(c). no_junk(plan) ≡ **dom** plan = ∪{**rng**(plan(hi))|hi:HI·hi ∈ **dom** 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

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

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

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

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

net_construction(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.

End of Lecture 2: ONTOLOGY

B Slide Table-of-Contents

Contents