Domain Science & Engineering

Lectures at BeiDa and ECNU[1]

Dines Bjørner

DTU Informatics

November 17, 2012: 09:45

[1] 22–23.11.2012 and 6–7.12.2012

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

Schedule

Summary

- By a **domain entity** we shall understand
  - *a manifest phenomenon, something that can be pointed to,*
  - *which has a form of*
    - *either permanent character, that is, an endurant,*
    - *or "fleeting" character, that is, a perdurant.*

Example: 1 Domain Entities. Example domain entities are

- a street segment (an endurant),
- insertion of a link (a perdurant).

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

- By a **quality** we shall understand
  - *a property, expressible as a proposition,*
  - *which is either satisfied by an entity or is not satisfied.*

Example: 2 Entity Qualities. Example entity qualities are

- length of a street segment (links): *has_length*,
- location of a street segments (links): *has_location*,
- identities of the street intersections connected to a link: *has_mereology*,
- type of arguments of the insertion function: *has_argument*,
- type of result of the insertion function: *has_result*.

- By a **domain** we shall understand
  - *a set of domain entities,*
  - *a set of qualities and*
  - *a mapping of domain entities into qualities.*

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

- By **domain analysis** we shall understand
  - *the principled use of a set of techniques and tools*
  - *for identifying*
    - *domain entities,*
    - *qualities and*
    - *mappings between them.*

- By **domain description** we shall understand
  - *the principled use of a set of techniques and tools*
  - *for describing, informally and formally*
    - *domain entities,*
    - *qualities and*
    - *mappings between them.*

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

- In this seminar
  - we shall cover, in respective sections,
  - the techniques and tools of domain analysis and
  - the techniques and tools of domain description.

1. Lecture 1: An Example

- The main example presents a terse narrative and formalisation of a road traffic domain.
  - Since the example description conceptually covers also major aspects of
    - railroad nets,
    - shipping nets, and
    - air traffic nets,
  - we shall use such terms as hubs and links to stand for
    - road (or street) intersection and road (or street) segments,
    - train stations and rail lines,
    - harbours and shipping lanes, and
    - airports and air lanes.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.1. Parts
1.1.1. Root Sorts

- The domain,
  - the stepwise unfolding of
  - whose description is
  - to be exemplified,

  is that of a **composite traffic system**
  - with a road net,
  - with a fleet of vehicles
  - of whose individual position on the road net we can speak, that is, monitor.

1. We analyse the composite traffic system into
   a a composite road net,
   b a composite fleet (of vehicles), and
   c an atomic monitor.

type
1. Δ
1a. N
1b. F
1c. M

value
1a. **obs_**N: Δ → N
1b. **obs_**F: Δ → F
1c. **obs_**M: Δ → M

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.1.2. Sub-domain Sorts and Types

2. From the road net we can observe
   a. a composite part, HS, of road (i.e., street) intersections (hubs) and
   b. an composite part, LS, of road (i.e., street) segments (links).

type
2. HS, LS
value
2a. **obs_**HS: N → HS
2b. **obs_**LS: N → LS

3. From the fleet sub-domain, F, we observe a composite part, VS, of vehicles

type
3. VS
value
3. **obs_**VS: F → VS

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

4. From the composite sub-domain VS we observe
   a the composite part Vs, which we concretise as a set of vehicles
   b where vehicles, V, are considered atomic.

type
4a. Vs = V**-set**
4b. V
value
4a. **obs_**Vs: VS → V**-set**

- The "monitor" is considered atomic; it is an abstraction of the fact that
  - we can speak of the positions of each and every vehicle on the net
  - without assuming that we can indeed pin point these positions
  - by means of for example sensors.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.1.3. Further Sub-domain Sorts and Types

- We now analyse the sub-domains of HS and LS.

5. From the hubs aggregate we decide to observe
   a the concrete type of a set of hubs,
   b where hubs are considered atomic; and
6. from the links aggregate we decide to observe
   a the concrete type of a set of links,
   b where links are considered atomic;

type
5a. Hs = H-**set**
6a. Ls = L-**set**
5b. H
6b. L
value
5. **obs_**Hs: HS → H-**set**
6. **obs_**Ls: LS → L-**set**

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

- We have no composite parts left to further analyse into parts
  - whether they be again composite
  - or atomic.
- That is,
  - at various, what we shall refer to as, **domain indexes**
  - we have discovered the following part types:
    - $\langle\Delta\rangle$: N, F, M
    - $\langle\Delta, N\rangle$: HS, LS
    - $\langle\Delta, F\rangle$: VS
    - $\langle\Delta, HS\rangle$: Hs, H
    - $\langle\Delta, LS\rangle$: Ls, L
    - $\langle\Delta, VS\rangle$: Vs, V
  - Thus we have ended up with atomic parts.

1.2. Properties

- Parts are distinguished by their properties:
  - the types and
  - the values

  of these.
- We consider three kinds of properties:
  - unique identifiers,
  - mereology and
  - attributes.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.2.1. Unique Identifications

7. We decide the following:
   a each hub has a unique hub identifier,
   b each link has a unique link identifier and
   c each vehicle has a unique vehicle identifier.

type
7a. HI
7b. LI
7c. VI
value
7a. **uid**_H: H → HI
7b. **uid**_L: L → LI
7c. **uid**_V: V → VI

1.2.2. Mereology

1.2.2.1 Road Net Mereology

- By *mereology* we mean the study, knowledge and practice of understanding parts and part relations.

8. Each link is connected to exactly two hubs, that is,
   a from each link we can observe its mereology, that is, the identities of these two distinct hubs,
   b and these hubs must be of the net of the link;
9. and each hub is connected to zero, one or more links, that is,
   a from each hub we can observe its mereology, that is, the identities of these links,
   b and these links must be of the net of the hub.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

value

8a. **mereo_**L: L → HI-**set**, **axiom** ∀ l:L·**card** **mereo_**L(l)=2

axiom

8b. ∀ n:N,l:L,hi:HI · l ∈ **obs_**Ls(**obs_**LS(n)) ∧ hi ∈ **mereo_**L(l)

8b. ⇒ ∃ h:H·h ∈ **obs_**Hs(**obs_**HS(n))∧**uid_**H(h)=hi

value

9a. **mereo_**H: H → LI-**set**

axiom

9b. ∀ n:N,h:H,li:LI · h ∈ **obs_**Hs(**obs_**HS(n)) ∧ li ∈ **mereo_**H(h)

9b. ⇒ ∃ l:L·l ∈ **obs_**Ls(**obs_**LS(n))∧**uid_**L(l)=li

1.2.2.2 Fleet of Vehicles Mereology

- In the traffic system that we are building up
  - there are no relations to be expressed between vehicles,
  - only between vehicles and the (single and only) monitor.
- Thus there is no mereology needed for vehicles.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.2.3. Attributes

- We shall model attributes of
  - links,
  - hubs and
  - vehicles.
- The composite parts,
  - aggregations of hubs, HS and Hs,
  - aggregations of links, LS and Ls and
  - aggregations of vehicles, VS and Vs,

  also have attributes, but we shall omit modelling them here.

1.2.3.1 Attributes of Links

10. The following are attributes of links.
    a Link states, lσ:LΣ, which we model as possibly empty sets of pairs of distinct identifiers of the connected hubs.
        • A link state expresses the directions that are open to traffic across a link.
    b Link state spaces, lω:LΩ which we model as the set of link states.
        • A link state space expresses the states that a link may attain across time.
    c Further link attributes are length, location, etcetera.

- Link states are usually dynamic attributes
- whereas
    - link state spaces,
    - link length and
    - link location (usually some curvature rendition)

  are considered static attributes.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

type
10a. $L\Sigma = (HI \times HI)$**-set**
axiom
10a. $\forall\ l\sigma{:}L\Sigma \cdot 0 \leq$ **card** $l\sigma \leq 2$
value
10a. **attr_**$L\Sigma$: $L \rightarrow L\Sigma$
axiom
10a. $\forall\ l{:}L \cdot$ **let** $\{hi,hi'\}=$**mereo_**$L(l)$ **in** **attr_**$L\Sigma(l) \subseteq \{(hi,hi'),(hi',hi)\}$ **end**
type
10b. $L\Omega = L\Sigma$**-set**
value
10b. **attr_**$L\Omega$: $L \rightarrow L\Omega$
axiom
10b. $\forall\ l{:}L \cdot$ **let** $\{hi,hi'\}=$**mereo_**$L(l)$ **in** **attr_**$L\Sigma(l) \in$ **attr_**$L\Omega(l)$ **end**
type
10c. LOC, LEN, ...
value
10c. **attr_**LOC: $L \rightarrow$ LOC, **attr_**LEN: $L \rightarrow$ LEN, ...

1.2.3.2 Attributes of Hubs

11. The following are attributes of hubs:
    a Hub states, $\mathsf{h}\sigma{:}\mathsf{H}\Sigma$, which we model as possibly empty sets of pairs of identifiers of the connected links.
        • A hub state expresses the directions that are open to traffic across a hub.
    b Hub state spaces, $\mathsf{h}\omega{:}\mathsf{H}\Omega$ which we model as the set of hub states.
        • A hub state space expresses the states that a hub may attain across time.
    c Further hub attributes are location, etcetera.

- Hub states are usually dynamic attributes
- whereas
    - hub state spaces and
    - hub location

  are considered static attributes.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

type
11a. HΣ = (LI × LI)**-set**
value
11a. **attr_**HΣ: H → HΣ
axiom
11a. ∀ h:H · **attr_**HΣ(h)⊆{(li,li′)|li,li′:LI·{li,li′}⊆**mereo_**H(h)}
type
11b. HΩ = HΣ**-set**
value
11b. **attr_**HΩ: H → HΩ
axiom
11b. ∀ h:H · **attr_**HΣ(h) ∈ **attr_**HΩ(h)
type
11c. LOC, ...
value
11c. **attr_**LOC: L → LOC, ...

1.2.3.3 Attributes of Vehicles

12. Dynamic attributes of vehicles include
    a position
        i. at a hub (about to enter the hub — referred to by the link it is coming from, the hub it is at and the link it is going to, all referred to by their unique identifiers or
        ii. some fraction "down" a link (moving in the direction from a from hub to a to hub — referred to by their unique identifiers)
        iii. where we model fraction as a real between 0 and 1 included.
    b velocity, acceleration, etcetera.
13. All these vehicle attributes can be observed.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

type

108a. VP = atH | onL

108(a)i. atH :: fli:LI × hi:HI × tli:LI

108(a)ii. onL :: fhi:HI × li:LI × frac:FRAC × thi:HI

108(a)iii. FRAC = **Real**, **axiom** ∀ frac:FRAC · 0 ≤ frac ≤ 1

108b. VEL, ACC, ...

value

13. **attr_**VP:V→VP, **attr_**onL:V→onL, **attr_**atH:V→atH

13. **attr_**VEL:V→VEL, **attr_**ACC:V→ACC

1.2.3.4 Vehicle Positions

14. Given a net, n:N, we can define the possibly infinite set of potential vehicle positions on that net, vps(n).
    a vps(n) is expressed in terms of the links and hubs of the net.
    b vps(n) is the
    c union of two sets:
        i. the potentially[2] infinite set of "on link" positions
        ii. for all links of the net
    and
        i. the finite set of "at hub" positions
        ii. for all hubs in the net.

[2] The 'potentiality' arises from the nature of FRAC. If fractions are chosen as, for example, 1/5'th, 2/5'th, ..., 4/5'th, then there are only a finite number of "on link" vehicle positions. If instead fraction are arbitrary infinitesimal quantities, then there are infinitely many such.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

value

14. vps: N → VP**-infset**
14b. vps(n) ≡
14a. **let** ls=**obs_**Ls(**obs_**LS(n)), hs=**obs_**Hs(**obs_**HS(n)) **in**
14(c)i. { onL(fhi,uid(l),f,thi) | fhi,thi:HI,l:L,f:FRAC ·
14(c)ii. l ∈ ls ∧ {fhi,thi}=**mereo_**L(l) }
14c. ∪
14(c)i. { atH(fli,**uid_**H(h),tli) | fli,tli:LI,h:H ·
14(c)ii. h ∈ hs ∧ {fli,tli}⊆**mereo_**H(h) }
14a. **end**

- Given a net and a finite set of vehicles
  - we can distribute these over the net, i.e., assign initial vehicle positions,
  - so that no two vehicles "occupy" the same position, i.e., are "crashed" !
- Let us call the non-deterministic assignment function, i.e., a relation, for vpr.

15. vpm:VPM is a bijective map from vehicle identifiers to (distinct) vehicle positions.
16. vpr has the obvious signature.
17. vpr(vs)(n) is defined in terms of
18. a non-deterministic selection, vpa, of vehicle positions, and
19. a non-deterministic assignment of these vehicle positions to vehicle identifiers —
20. being the resulting distribution.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

type

15. VPM′ = VI $\overrightarrow{m}$ VP
15. VPM = {| vpm:VPM′ · **card dom** vpm = **card rng** vpm |}

value

16. vpr: V**-set** × N → VMP
17. vpr(vs)(n) ≡
18. **let** vpa:VP**-set** · vpa ⊆ vps(vs)(n) ∧ **card** vpa = vard vs **in**
19. **let** vpm:VPM · **dom** vpm = vps ∧ **rng** vpm = vpa **in**
20. vpm **end end**

1.3. Definitions of Auxiliary Functions

21. From a net we can extract all its link identifiers.
22. From a net we can extract all its hub identifiers.

value

21. xtr_LIs: N → LI-**set**
21. xtr_LIs(n) ≡ {**uid**_L(l)|l:L·l ∈ **obs**_Ls(**obs**_LS(n))}
22. xtr_HIs: N → HI-**set**
22. xtr_HIs(n) ≡ {**uid**_H(l)|h:H·h ∈ **obs**_Hs(**obs**_HS(n))}

23. Given a link identifier and a net get the link with that identifier in the net.
24. Given a hub identifier and a net get the hub with that identifier in the net.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

value

26. get_H: HI → N $\xrightarrow{\sim}$ H
26. get_H(hi)(n) ≡ ι h:H·h ∈ **obs**_Hs(**obs**_HS(n))∧**uid**_H(h)=hi
26. **pre**: hi ∈ xtr_HIs(n)
26a. get_L: LI → N $\xrightarrow{\sim}$ L
26a. get_L(li)(n) ≡ ι l:L·l ∈ **obs**_Ls(**obs**_LS(n))∧**uid**_L(l)=li
26a. **pre**: hl ∈ xtr_LIs(n)

- The ι a:A·$\mathcal{P}$(a) expression
  - yields the unique value a:A
  - which satisfies the predicate $\mathcal{P}$(a).
  - If none, or more than one exists then the function is undefined.

1.4. Some Derived Traffic System Concepts

1.4.1. Maps

25. A road map is an abstraction of a road net. We define one model of maps below.
    a A road map, RM, is a finite definition set function, M, (a specification language map) from
        - hub identifiers (the source hub)
        - to (such finite definition set) functions
        - from link identifiers
        - to hub identifiers (the target hub).

type

25a. $\mathrm{RM}' = \mathrm{HI} \xrightarrow{m} (\mathrm{LI} \xrightarrow{m} \mathrm{HI})$

- If a hub identifier in the source or an rm:RM maps into the empty map then the "corresponding" hub is "isolated": has no links emanating from it.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

26. These road maps are subject to a well-formedness criterion.
    a The target hubs must be defined also as source hubs.
    b If a link is defined from source hub (referred to by its identifier) **shi** via link **li** to a target hub **thi**, then, vice versa, link **li** is also defined from source **thi** to target **shi**.

type
26. RM = {| rm:RM′ · wf_RM(rm) |}
value
26. wf_RM: RM′ → **Bool**
26. wf_RM(rm) ≡
26a. ∪ { **rng**(rm(hi))|hi:HI·hi ∈ **dom** rm } ⊆ **dom** rm
26b. ∧ ∀ shi:HI·shi ∈ **dom** rm ⇒
26b. ∀ li:LI · li ∈ **dom** rm(shi) ⇒
26b. li ∈ **dom** rm((rm(shi))(li)) ∧ (rm((rm(shi))(li)))(li)=shi

27. Given a road net, n, one can derive "its" road map.
    a Let hs and ls be the hubs and links, respectively of the net n.
    b Every hub with no links emanating from it is mapped into the empty map.
    c For every link identifier uid_L(l) of links, l, of ls and every hub identifier, hi, in the mereology of l
    d hi is mapped into a map from uid_L(l) into hi'
    e where hi' is the other hub identifier of the mereology of l.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

value

27. derive_RM: N → RM
27. derive_RM(n) ≡
27a. **let** hs = **obs_**Hs(**obs_**HS(n)), ls = **obs_**Ls(**obs_**LS(n)) **in**
27b. [hi ↦ [ ] | hi:HI · ∃ h:H · h ∈ hs ∧ **mereo_**H(h) = {}] ∪
27d. [hi ↦ [**uid_**L(l) ↦ hi′
27e. | hi′:HI · hi′ = **mereo_**L(l)\{hi}]
27c. | l:L,hi:HI · l ∈ ls ∧ hi ∈ **mereo_**L(l)] **end**

- **Theorem:** If the road net, n, is well-formed then wf_RM(derive_RM(n)).

1.4.2. Traffic Routes

28. A traffic route, tr, is an alternating sequence of hub and link identifiers such that
    a li:LI is in the mereology of the hub, h:H, identified by hi:HI, the predecessor of li:LI in route r, and
    b hi':HI, which follows li:LI in route r, is different from hi, and is in the mereology of the link identified by li.

type
28. R′ = (HI|LI)*
28. R = {| r:R′ • ∃ n:N • wf_R(r)(n) |}
value
28. wf_R: R′ → N → **Bool**
28. wf_R(r)(n) ≡
28. ∀ i:**Nat** • {i,i+1}⊆**inds** r ⇒
28a. **is_**HI(r(i)) ⇒ **is_**LI(r(i+1)) ∧ r(i+1) ∈ **mereo_**H(get_H(r(i))(n)),
28b. **is_**LI(r(i)) ⇒ **is_**HI(r(i+1)) ∧ r(i+1) ∈ **mereo_**L(get_L(r(i))(n))

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

29. From a well-formed road map (i.e., a road net) we can generate the possibly infinite set of all routes through the net.
    a **Basis Clauses:**
        i. The empty sequence of identifiers is a route.
        ii. The one element sequences of link and hub identifiers of links and hubs of a road map (i.e., a road net) are routes.
        iii. If hi maps into some li in rm then $\langle \mathsf{hi},\mathsf{li} \rangle$ and $\langle \mathsf{li},\mathsf{hi} \rangle$ are routes of the road map (i.e., of the road net).
    b **Induction Clause:**
        i. Let $\mathsf{r}^\frown\langle \mathsf{i} \rangle$ and $\langle \mathsf{i}' \rangle^\frown\mathsf{r}'$ be two routes of the road map.
        ii. If the identifiers i and i' are identical, then $\mathsf{r}^\frown\langle \mathsf{i} \rangle^\frown\mathsf{r}'$ is a route.
    c **Extremal Clause:**
        i. Only such routes that can be formed from a finite number of applications of the above clauses are routes.

value

29. gen_routes: M → Routes-**infset**
29. gen_routes(m) ≡
29(a)i. **let** rs = {⟨⟩}
29(a)ii. ∪ {⟨li,hi⟩,⟨hi,li⟩|li:LI,hi:HI•...}
29(b)i. ∪ {**let** r⌢⟨li⟩,⟨li′⟩⌢r′:R • {r⌢⟨li⟩,⟨li′⟩⌢r′}⊆rs,
29(b)i. r″⌢⟨hi⟩,⟨hi′⟩⌢r‴:R • {r″⌢⟨hi⟩,⟨hi′⟩⌢r‴}⊆rs **in**
29(b)ii. r⌢⟨li⟩⌢r′,r″⌢⟨hi⟩⌢r‴ **end**} **in**
29(c)i. rs **end**

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.4.2.1 Circular Routes

30. A route is circular if the same identifier occurs more than once.

value

30. is_circular_route: R → **Bool**
30. is_circular_route(r) ≡ ∃ i,j:**Nat** · {i,j}⊆**inds** r ∧ i≠j ⇒ r(i)=r(j)

1.4.2.2 Connected Road Nets

31. A road net is connected if there is a route from any hub (or any link) to any other hub or link in the net.

```
31.  is_conn_N: N → Bool
31.  is_conn_N(n) ≡
31.      let m = derive_RM(n) in
31.      let rs = gen_routes(m) in
31.      ∀ i,i′:(LI|HI) · {i,i′}⊆xtr_LIs(n)∪ xtr_HIs(n)
31.          ∃ r:R · r ∈ rs ∧ r(1)=i ∧ r(len r)=i′ end end
```


© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.4.2.3 Set of Connected Nets of a Net

32. The set, **cns**, of connected nets of a net, **n**, is

    a the smallest set of connected nets, **cns**,

    b whose hubs and links together "span" those of the net **n**.

value

32. conn_Ns: N → N-**set**
32. conn_Ns(n) **as** cns
32a. **pre**: **true**
32b. **post**: conn_spans_HsLs(n)(cns)
32a. ∧ ∼∃ kns:N-**set** · **card** kns < **card** cns
32a. ∧ conn_spans_HsLs(n)(kns)

32b. conn_spans_HsLs: N → N → **Bool**
32b. conn_spans_HsLs(n)(cns) ≡
32b. ∀ cn:N·cn ∈ cns ⇒ is_connected_N(n)(cn)
32b. ∧ **let** (hs,ls) = (**obs_**Hs(**obs_**HS(n)),**obs_**Ls(**obs_**LS(n))),
32b. chs = ∪{**obs_**Hs(**obs_**HS(cn))|cn ∈ cns},
32b. cls = ∪{**obs_**Ls(**obs_**LS(cn))|cn ∈ cns} **in**
32b. hs = chs ∧ ls = cls **end**

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.4.2.4 Route Length

33. The length attributes of links can be
    a added and subtracted,
    b multiplied by reals to obtain lengths,
    c divided to obtain fractions,
    d compared as to whether one is shorter than another, etc., and
    e there is a "zero length" designator.

value

33a. $+,- : \text{LEN} \times \text{LEN} \to \text{LEN}$
33b. $* : \text{LEN} \times \textbf{Real} \to \text{LEN}$
33c. $/ : \text{LEN} \times \text{LEN} \to \textbf{Real}$
33d. $<,\leq,=,\neq,\geq,> : \text{LEN} \times \text{LEN} \to \textbf{Bool}$
33e. $\ell_0 : \text{LEN}$

34. One can calculate the length of a route.

value

34. length: R → N → LEN
34. length(r)(n) ≡
34. **case** r **of**:
34. ⟨⟩ → ℓ_0,
34. ⟨si⟩⌢r′ →
34. is_LI(si)→**attr**_LEN(get_L(si)(n))+length(r′)(n)
34. is_HI(si)→length(r′)(n)
34. **end**

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.4.2.5 Shortest Routes

35. There is a predicate, is_R, which,
    a given a net and two distinct hub identifiers of the net,
    b tests whether there is a route between these.

value

35. is_R: N → (HI×HI) → **Bool**
35. is_R(n)(fhi,thi) ≡
35a. fhi ≠ thi ∧ {fht,thi}⊆xtr_HIs(n)
35b. ∧ ∃ r:R · r ∈ routes(n) ∧ **hd** r = fhi ∧ r(**len** r) = thi

36. The shortest between two given hub identifiers

    a is an acyclic route, r,

    b whose first and last elements are the two given hub identifiers

    c and such that there is no route, r′ which is shorter.

value

36. shortest_route: N → (HI×HI) → R
36a. shortest_route(n)(fhi,thi) **as** r
36b. **pre**: pre_shortest_route(n)(fhi,thi)
36c. **post**: pos__shortest_route(n)(r)(fhi,thi)

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

36b. pre_shortest_route: N → (HI×HI) → **Bool**
36b. pre_shortest_route(n)(fhi,thi) ≡
36b. is_R(n)(fhi,thi) ∧ fhi≠thi ∧ {fhi,thi}⊂xtr_HIs(n)

36c. pos_shortest_route: N → R → (HI×HI) → **Bool**
36c. pos_shortest_route(n)(r)(fhi,thi) ≡
36c. r ∈ routes(n)
36c. ∧ ∼∃ r′:R · r′ ∈ routes(n) ∧ length(r′) < length(r)

1.5. States

- There are different notions of state. In our example these are some of the states:
  - the road net composition of hubs and links;
  - the state of a link, or a hub; and
  - the vehicle position.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.6. Actions

- An action is what happens when a function invocation changes, or potentially changes a state.
- Examples of traffic system actions are:
  - insertion of hubs,
  - insertion of links,
  - removal of hubs,
  - removal of links,
  - setting of hub state ($\mathsf{h}\sigma$),
  - setting of link state ($\mathsf{l}\sigma$),
  - moving a vehicle along a link,
  - moving a vehicle from a link to a hub and
  - moving a vehicle from a hub to a link.

37. The **insert** action applies to a net and a hub and conditionally yields an updated net.
    a The condition is that there must not be a hub in the "argument" net with the same unique hub identifier as that of the hub to be inserted and
    b the hub to be inserted does not initially designate links with which it is to be connected.
    c The updated net contains all the hubs of the initial net "plus" the new hub.
    d and the same links.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

value

83. ins_H: N → H $\xrightarrow{\sim}$ N
83. ins_H(n)(h) **as** n′, **pre**: pre_ins_H(n)(h), **post**: post_ins_H(n)(h)

83a. pre_ins_H(n)(h) ≡
83a. ∼∃ h′:H · h′ ∈ **obs**_Hs(n) ∧ **uid**_HI(h)=**uid**_HI(h′)
83b. ∧ **mereo**_H(h) = {}

83c. post_ins_H(n)(h)(n′) ≡
83c. **obs**_Hs(n) ∪ {h} = **obs**_Hs(n′)
83d. ∧ **obs**_Ls(n) = **obs**_Ls(n′)

1.7. Events

- By an **event** we understand
  - a state change
  - resulting indirectly from an unexpected application of a function,
  - that is, that function was performed "surreptitiously".
- Events can be characterised by a pair of (before and after) states, a predicate over these and, optionally, a **time** or **time interval**.
- Events are thus like actions:
  - change states,
  - but are usually
    - either caused by "previous" actions,
    - or caused by "an outside action".

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

38. Link disappearance is expressed as a predicate on the "before" and "after" states of the net. The predicate identifies the "missing" ℓink (!).
39. Before the disappearance of link ℓ in net n
    a the hubs h' and h'' connected to link ℓ
    b were connected to links identified by $\{l'_1, l'_2, \ldots, l'_p\}$ respectively $\{l''_1, l''_2, \ldots, l''_q\}$
    c where, for example, l'_i, l''_j are the same and equal to uid_$\Pi(\ell)$.

84. link_dis: N × N → **Bool**
84. link_dis(n,n′) ≡
84. ∃ ℓ:L · pre_link_dis(n,ℓ) ⇒ post_link_dis(n,ℓ,n′)
85. pre_link_dis: N × L → **Bool**
85. pre_link_dis(n,ℓ) ≡ ℓ ∈ **obs_**Ls(n)

40. After link ℓ disappearance there are instead
    a two separate links, ℓ_i and ℓ_j, "truncations" of ℓ
    b and two new hubs h''' and h''''
    c such that ℓ_i connects h' and h''' and
    d ℓ_j connects h'' and h'''';
    e Existing hubs h' and h'' now have mereology
        i. $\{l'_1, l'_2, \ldots, l'_p\} \setminus \{\mathsf{uid\_\Pi}(\ell)\} \cup \{\mathsf{uid\_\Pi}(\ell_i)\}$ respectively
        ii. $\{l''_1, l''_2, \ldots, l''_q\} \setminus \{\mathsf{uid\_\Pi}(\ell)\} \cup \{\mathsf{uid\_\Pi}(\ell_j)\}$
41. All other hubs and links of n are unaffected.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

42. We shall "explain" *link disappearance* as the combined, instantaneous effect of
    a. first a **remove link** "event" where the **removed link** connected **hub**s hi_j and hi_k;
    b. then the **insert**ion of two new, "fresh" **hub**s, h_α and h_β;
    c. "followed" by the **insert**ion of two new, "fresh" links $\mathsf{l}_{j\alpha}$ and $\mathsf{l}_{k\beta}$ such that
        i. $\mathsf{l}_{j\alpha}$ connects hi_j and h_α and
        ii. $\mathsf{l}_{k\beta}$ connects hi_k and $\mathsf{h}_{k\beta}$

value

88. post_link_dis(n,ℓ,n$'$) ≡
88. **let** h_a,h_b:H ·
88. **let** {li_a,li_b}=**mereo_**L(ℓ) **in**
88. (get_H(li_a)(n),get_H(li_b)(n)) **end in**
88a. **let** n$''$ = rem_L(n)(**uid_**L(ℓ)) **in**
88b. **let** h$_\alpha$,h$_\beta$:H · {h$_\alpha$,h$_\beta$}∩**obs_**Hs(n)={} **in**
88b. **let** n$'''$ = ins_H(n$''$)(h$_\alpha$) **in**
88b. **let** n$''''$ = ins_H(n$'''$)(h$_\beta$) **in**
88c. **let** l$_{j\alpha}$,l$_{k\beta}$:L · {l$_{j\alpha}$,l$_{k\beta}$}∩**obs_**Ls(n)={}
88c. ∧ **mereo_**L(l$_{j\alpha}$) = {**uid_**H(h_a),**uid_**H(h$_\alpha$)}
88c. ∧ **mereo_**L(l$_{k\beta}$) = {**uid_**H(h_b),**uid_**H(h$_\beta$)} **in**
88(c)i. **let** n$'''''$ = ins_L(n$''''$)(l$_{j\alpha}$) **in**
88(c)ii. n$'$ = ins_L(n$'''''$)(l$_{k\beta}$) **end end end end end end end**

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.8. Behaviours
1.8.1. Traffic
1.8.1.1 Continuous Traffic

- For the road traffic system
  - ◈ perhaps the most significant example of a behaviour
  - ◈ is that of its traffic
    43. the continuous time varying discrete positions of vehicles, vp:VP[3],
    44. where time is taken as a dense set of points.

type

90. $\mathbb{cT}$
89. $\mathrm{cRTF} = \mathbb{cT} \rightarrow (\mathrm{V} \xrightarrow[m]{} \mathrm{VP})$

[3] For VP see Item 108a on Slide 243.

1.8.1.2 Discrete Traffic

- We shall model, not continuous time varying traffic, but

45. discrete time varying discrete positions of vehicles,
46. where time can be considered a set of linearly ordered points.

92. $\mathbb{dT}$
91. $\mathbb{dRTF} = \mathbb{dT} \xrightarrow{m} (\mathbb{V} \xrightarrow{m} \mathbb{VP})$

47. The road traffic that we shall model is, however, of vehicles referred to by their unique identifiers.

type

93. $\mathbb{RTF} = \mathbb{dT} \xrightarrow{m} (\mathbb{VI} \xrightarrow{m} \mathbb{VP})$

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.8.1.3 Time: An Aside

- We shall take a rather simplistic view of time
  [wayne.d.blizard.90,mctaggart-t0,prior68,J.van.Benthem.Log

48. We consider **d**$\mathbb{T}$, or just $\mathbb{T}$, to stand for a totally ordered set of time points.
49. And we consider $\mathbb{TI}$ to stand for time intervals based on $\mathbb{T}$.
50. We postulate an infinitesimal small time interval δ.
51. $\mathbb{T}$, in our presentation, has lower and upper bounds.
52. We can compare times and we can compare time intervals.
53. And there are a number of "arithmetics-like" operations on times and time intervals.

type

94. $\mathbb{T}$
95. $\mathbb{TI}$

value

96. δ:$\mathbb{TI}$
97. $\mathbb{MIN}, \mathbb{MAX}$: $\mathbb{T} \rightarrow \mathbb{T}$
97. $<,\leq,=,\geq,>$: $(\mathbb{T}\times\mathbb{T})|(\mathbb{TI}\times\mathbb{TI}) \rightarrow$ **Bool**
98. $-$: $\mathbb{T}\times\mathbb{T} \rightarrow \mathbb{TI}$
99. $+$: $\mathbb{T}\times\mathbb{TI},\mathbb{TI}\times\mathbb{T} \rightarrow \mathbb{T}$
99. $-,+$: $\mathbb{TI}\times\mathbb{TI} \rightarrow \mathbb{TI}$
99. $*$: $\mathbb{TI}\times$**Real** $\rightarrow \mathbb{TI}$
99. $/$: $\mathbb{TI}\times\mathbb{TI} \rightarrow$ **Real**

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

54. We postulate a global **clock** behaviour which offers the current time.

55. We declare a channel **clk_ch**.

value

100. clock: $\mathbb{T} \rightarrow$ **out** clk_ch **Unit**

100. clock(t) $\equiv$ ... clk_ch!t ... clock(t $\prod$ t+δ)

channnel

101. clk_ch:$\mathbb{T}$

1.8.2. Globally Observable Parts

- There is given

56. a net, n:N,
57. a set of vehicles, vs:V-set, and
58. a monitor, m:M.

- The n:N, vs:V-set and m:M are observable from the road traffic system domain.

value

103. n:N = **obs_**N(Δ)
103. ls:L-set = **obs_**Ls(**obs_**LS(n)), hs:H-set = **obs_**Hs(**obs_**HS(n)),
103. lis:LI-set = {**uid_**L(l)|l:L·l ∈ ls}, his:HI-set = {**uid_**H(h)|h:H·h ∈ hs}
104. vs:V-set = **obs_**Vs(**obs_**VS(**obs_**F(Δ))), vis:V-set = {**uid_**V(v)|v:V·
105. m:**obs_**M(Δ)

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.8.3. Road Traffic System Behaviours

59. Thus we shall consider our road traffic system, **rts**, as
    a the concurrent behaviour of a number of vehicles and,
      to "observe", or, as we shall call it, to monitor their movements,
    b the **moni**tor behaviour, based on
    c the monitor and its unique identifier,
    d an initial vehicle position map, and
    e an initial starting time.

value

59c. mi:MI = **uid_**(m)

59d. vpm:VPM = vpr(vs)(n)

59e. t_0:T = clk_ch?

102. rts() =

102a. ‖ {veh(**uid_**V(v))(v)(vpm(**uid_**V(v)))|v:V·v ∈ vs}

102b. ‖ mon(mi)(m)([t_0 ↦ vpm])

- where the "extra" monitor argument
  - records the discrete road traffic, RTF,
  - initially set to the singleton map from an initial start time, t_0 to the initial assignment of vehicle positions.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.8.4. Channels

- In order for the monitor behaviour to assess the vehicle positions
  - these vehicles communicate their positions
  - to the monitor
  - via a vehicle to monitor channel.
- In order for the monitor to time-stamp these positions
  - it must be able to "read" a clock.

60. Thus we declare a set of channels indexed by the unique identifiers of vehicles and communicating vehicle positions.

channel

106. {vm_ch[mi,vi]|vi:VI·vi ∈ vis}:VP

1.8.5. Behaviour Signatures

61. The road traffic system behaviour, **rts**, takes no arguments; and "behaves", that is, continues forever.
62. The vehicle behaviours are indexed by the unique identifier, **uid_V(v):VI**, the vehicle part, **v:V** and the vehicle position; offers communication to the monitor behaviour; and behaves "forever".
63. The monitor behaviour takes monitor part, **m:M**, as argument and also the discrete road traffic, **drtf:dRTF**; the behaviour otherwise runs forever.

value

109. rts: **Unit** → **Unit**
110. veh: vi:VI → v:V → VP → **out** vm_ch[vi],mi:MI **Unit**
111. mon: mi:MI → m:M → dRTF → **in** {vm_ch[mi,vi]|vi:VI·vi ∈ vis},clk

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

1.8.6. The Vehicle Behaviour

64. A vehicle process
    - is indexed by the unique vehicle identifier vi:VI,
    - the vehicle "as such", v:V and
    - the vehicle position, vp:VPos.

    The vehicle process communicates
    - with the monitor process on channel vm[vi]
    - (sends, but receives no messages), and
    - otherwise evolves "in[de]finitely" (hence **Unit**).

65. We describe here an abstraction of the vehicle behaviour **at** a **Hub** (**hi**).
    a Either the vehicle remains at that hub informing the monitor,
    b or, internally non-deterministically,
        i. moves onto a link, **tli**, whose "next" hub, identified by **thi**, is obtained from the mereology of the link identified by **tli**;
        ii. informs the monitor, on channel **vm[vi]**, that it is now on the link identified by **tli**,
        iii. whereupon the vehicle resumes the vehicle behaviour positioned at the very beginning (**0**) of that link,
    c or, again internally non-deterministically,
    d the vehicle "disappears — off the radar" !

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

113. veh(vi)(v)(vp:atH(fli,hi,tli)) ≡
113a. vm_ch[mi,vi]!vp ; veh(vi)(v)(vp)
113b. ⌈⌉
113(b)i. **let** {hi′,thi}=**mereo_**L(get_L(tli)(n)) **in assert:** hi′=hi
113(b)ii. vm_ch[mi,vi]!onL(tli,hi,0,thi) ;
113(b)iii. veh(vi)(v)(onL(tli,hi,0,thi)) **end**
113c. ⌈⌉
113d. **stop**

66. We describe here an abstraction of the vehicle behaviour **on** a **L**ink (**ii**). Either
    a the vehicle remains at that link position informing the monitor,
    b or, internally non-deterministically,
    c if the vehicle's position on the link has not yet reached the hub,
        i. then the vehicle moves an arbitrary increment δ along the link informing the monitor of this, or
        ii. else, while obtaining a "next link" from the mereology of the hub (where that next link could very well be the same as the link the vehicle is about to leave),
            A. the vehicle informs the monitor that it is now at the hub identified by **thi**,
            B. whereupon the vehicle resumes the vehicle behaviour positioned at that hub.
67. or, internally non-deterministically,
68. the vehicle "disappears — off the radar" !

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

112. veh(vi)(v)(vp:onL(fhi,li,f,thi)) ≡
114a. vm_ch[mi,vi]!vp ; veh(vi)(v)(vp)
114b. ⌈⌉
114c. **if** f + δ<1
114(c)i. **then** vm_ch[mi,vi]!onL(fhi,li,f+δ,thi) ;
114(c)i. veh(vi)(v)(onL(fhi,li,f+δ,thi))
114(c)ii. **else let** li′:LI·li′ ∈ **mereo**_H(get_H(thi)(n)) **in**
114(c)iiA. vm_ch[mi,vi]!atH(li,thi,li′);
114(c)iiB. veh(vi)(v)(atH(li,thi,li′)) **end end**
115. ⌈⌉
116. **stop**

1.8.7. The Monitor Behaviour

69. The monitor behaviour evolves around the attributes of an own "state", m:M, a table of traces of vehicle positions, while accepting messages about vehicle positions and otherwise progressing "in[de]finitely".
70. Either the monitor "does own work"
71. or, internally non-deterministically accepts messages from vehicles.
    a. A vehicle position message, vp, may arrive from the vehicle identified by vi.
    b. That message is appended to that vehicle's movement trace,
    c. whereupon the monitor resumes its behaviour —
    d. where the communicating vehicles range over all identified vehicles.

© Dines Bjørner 2012, DTU Informatics, Techn.Univ.of Denmark – November 17, 2012: 09:45

117. mon(mi)(m)(rtf) ≡
118. mon(mi)(own_mon_work(m))(rtf)
119. ⌈⌉
119a. [] { **let** ((vi,vp),t) = (vm_ch[mi,vi]?,clk_ch?) **in**
119b. **let** rtf′ = rtf † [t ↦ rtf(max **dom** rtf) † [vi ↦ vp]] **in**
119c. mon(mi)(m)(rtf′) **end**
119d. **end** | vi:VI · vi ∈ vis }

118. own_mon_work: M → dRTF → M

- We do not describe the clock behaviour by other than stating that it continually offers the current time on channel clkm_ch. ■