3. Base Urban Planning

Second Lecture

- **Agenda:**

3.1. Urban Planning Information Categories

3.1.1. "Input"

- Among the arguments of urban planning are

1 information, bTSU, about *the urban space,* the demo-geographic area subject to planning: its geodetic "make-up", its geotechnical and meteorological properties, etc.[2]

2 related, but not geographic, information, bAUX[iliary][3],

3 and some requirements, bREQ[uirements].

type
1 bTSU
2 bAUX
3 bREQ

[2] We shall not include information about the use of the existing land. We refer to Slide 49 for how we handle such information.
[3] Auxiliary: giving help or support.

3.1.2. "Output"

Among results of urban planning are

4 "the plan" (or "plans"), bPLA[ns],

5 and possibly some other ancillary[4] documents, bANC[illerary].

```
type
4  bPLA
5  bANC
```

[4]Ancillary: providing necessary support to the main work

3.2. The Iterative Nature of Urban Planning

- We take it that urban planning proceeds in “cycles”:

6 In each cycle the base urban planning function, base_up_fct, is applied to an input argument triple, (b_tus,b_aux,b_req):(bTUS×bAUX×bREQ):bTRI, of “fresh” geodetic/geotechnical/meteorological (etc.), auxiliary and requirements information.

type

6 $\mathsf{bTRI} = \mathsf{bTUS} \times \mathsf{bAUX} \times \mathsf{bREQ}$

7 Each cycle, that is, each application of base_up_fct, results in a “most recent”, not necessarily “final”, plan and ancillary information, (b_pla,b_anc):bPLA×bANC:bRES.

type

7 bRES = bPLA × bANC

8 But, to “drive” the urban planning process, base_up_beh, towards “final”, that is, an adequately satisfactory plan etc., the urban planning function, base_up_fct,

- *need also be provided with the results of the previous iteration’s result* —
- which we take to be a (“quintuplet”) pair
- of an (i.e., the “previous”) “input” triple
- and the previous result pair.

type

8 bQUI = bTRI × bRES

3.3. Initialisation

Urban planning proceeds in iterating from initial

9 urban space, auxiliary and requirements information, as well as

10 (usually "empty") plans and ancillaries.

We extend the notion of initial values to

11 triplet arguments,

12 result pairs, and

13 "quintuplet" argument/result pairs.

towards such results (plans and ancillaries) that are deemed satisfactory.

```
value
9   b_tus_init: bTUS,
9   b_aux_init:bAUX,
9   b_req_init:bREQ
10  b_pla_init: bPLA,
10  b_anc_init:bANC
11  b_tri_init: bTRI = (b_tus_init,b_aux_init,b_req_init)
11                  assert: b_tri_fit(b_tri_init,b_tri_init)
12  b_res_init: bRES = (b_pla_init,b_anc_init)
13  b_qui_init: bQUI = (b_tri_init,b_res_init)
```

- We refer to Item 23 on Slide 56 for an explanation of the b_tri_fit predicate.

3.3.1. Existing versus Evolving Plans

- The quintuplet "feedback",
  - which includes a 'plan' component,
  - secures that possibly pre-existing plans
  - are included, as initialised components of the plan results.[5]
- The iterative nature of urban planning
  - thus allows for stepwise urban re-development,
  - from existing "urban land-scapes"
  - via mixed "previous" and "future" land-scapes
  - to the final urban development plan.

[5]We refer to Item 1 on Slide 42 and footnote 2 on Slide 42.

3.4. A Simple Functional Form

14 The base urban planning *function*, base_up_fct, thus applies to

- (i) a "most recent" triplet of urban space, auxiliary and requirements information, and to
- (ii) a "past quintuplet", a resumption[6], that is, pair of urban space, auxiliary and requirements information as well as plan and ancillary information and yields such a resumption "quintuplet" pair of a triplet and a pair.

15 The application of base_up_fct to such arguments, i.e.,

- base_up_fct(b_tus,b_aux,b_req)(b_qui)
- yields a "quintuplet" result, a resumption,
- b_qui:((b_tus′,b_aux′,b_req′),(b_pla,b_anc)).

[6]Resumption: like a repetition, a continuation

We "explain" the relations between "input" arguments and "output" (**as**) results:

16 The "input" argument b_tri
   is "carried forward", b_tri′ (=b_tri),
   to be redeposited as part of the result.

17 The main part of the result, (b_pla,b_anc),
   is related, $\mathcal{P}$, to the input argument
   including the previous "result", the resumption.

```
14   base_up_fct: bTRI → bQUI → bQUI
15   base_up_fct(b_tri)(b_qui) as (b_tri′,(b_pla,b_anc))
16        b_tri = b_tri′ ∧
17        $\mathcal{P}_{\mathsf{base}}$(b_tri)(b_qui)(b_pla,b_anc)
```

- For the time being we shall leave the base urban planning function, base_up_fct, that is, $\mathcal{P}_{\mathsf{base}}$, uninterpreted.

3.5. Oracles and Repositories

- Oracles are simple behaviours
  - that *offers* information to other behaviours.
- Repositories are simple behaviours
  - that *store* information from other behaviours
  - and *offers* stored information to other behaviours.

3.5.1. The Base 'Input' Oracle

- An urban planning oracle, when so requested, will select some information – usually in some non-deterministic fashion, and usually subject to some constraint – and present this information to the requestor, i.e., an urban planning behaviour.
- In this section we shall deal with one specific oracle, b_tri_beh:
  - ◈ one that "assembles" triplets, b_tri,
    - ◎ of urban space, b_tus:bTUS,
    - ◎ auxiliary, b_aux:bAUX, and
    - ◎ requirements, b_req:bREQ,

    information
  - ◈ to requesting behaviours.

We introduce a pair of specification components:

18 a *channel*, b_tri_ch,

- over which a base urban planning behaviour, base_up_beh,
- offers to receive triplets, b_tri:bTRI,
- from an oracle, b_tri_beh,

19 and an *oracle*, b_tri_beh,
which "remembers" its most recently communicated triplet[7].

channel
18 b_tri_ch:bTRI
value
19 b_tri_beh: bTRI → **out** b_tri_ch **Unit**

[7]The oracle is initialised with b_tri_beh(geo_init,b_aux_init,b_req_init).

20 The oracle assembles (b_tri′:bTRI),
a base triplet which satisfies a predicate b_tri_fit(b_tri,b_tri′).

21 That triplet is offered,
b_tri_ch ! b_tri′,
to the base urban behaviour –

22 whereupon the oracle resumes being the oracle, now, however,
with the recently assembled base triplet as its resumption.

```
19   b_tri_beh(b_tri) ≡
20       let b_tri′:bTRI · b_tri_fit(b_tri,b_tri′) in
21       b_tri_ch ! b_tri′ ;
22       b_tri_beh(b_tri′)
19       end
24   pre: b_tri_fit(b_tri,b_tri)
```

23 The fitness predicate, b_tri_fit(b_tri,b_tri′), checks whether a "newly" assembled base triplet, b_tri′, stands in some suitable[8] relation $\mathcal{P}$(b_tri,b_tri′) to a a similar (f.ex., "earlier") base triplet, b_tri.

24 The fitness predicate holds for b_tri_fit(b_tri,b_tri).

```
23   b_tri_fit: bTRI × bTRI → Bool
23   b_tri_fit(tri,tri′) ≡ 𝒫(b_tri,b_tri′)
```

25 The oracle, b_tri_beh, is initialised with an initial triplet value b_tri_init, cf. formula Item 11 on Slide 47.

```
23   b_tri_beh(b_tri_init): assert: b_tri_fit(b_tri_init,b_tri_init)
```

[8]– to be defined for each specific urban planning project

3.5.2. The Base Resumption Repository

- The "quintuplet" pair of
  - an "input" triple
  - and a result pair,
  - b_qui:(b_tri:bTRI,(b_pla:bPLA,b_anc:bANC))
- is thought of as residing in a *repository* behaviour, b_qui_beh,
  - which "receives" (b_qui_ch?) "quintuplets"
    from the urban planning behaviour,
  - or "offers" (b_qui_ch!b_qui) such to the urban planning behaviour.

26 There is therefore a channel, b_qui_ch, between the urban planning behaviour and the “quintuplet” repository behaviour,

27 b_qui_beh.

28 It either

29 accepts or

30 offers quintuplets.

```
channel
26    b_qui_ch:bQUI
value
27    b_qui_beh: bQUI → in,out b_qui_ch  Unit
27    b_qui_beh(b_qui) ≡
29        b_qui_beh(b_qui_ch?)
28        ⌈⌉
30        b_qui_ch!(b_qui) ; b_qui_beh(b_qui)
```

3.6. A Simple Behavioural Form

- Urban planning, however, is a time-consuming “affair”.
- So we model it as a behaviour.

31 The base_up_beh_0[9] behaviour

- takes no argument, hence the left signature element: **Unit**,
- avails itself of the input channel for obtaining
  - ◈ proper input, b_tri,
  - ◈ and b_qui,

  for the base urban function, base_up_fct,
- and output channel, for depositing a resumption, b_qui′,
- and (then) “goes on forever”, as indicated by the right signature element: **Unit**.

31 base_up_beh_0: **Unit** → **in** b_tri_ch **out** b_qui_ch **Unit**

[9] As there will be several versions, from simple towards more elaborate, of the base_up_beh behaviour, we index them.

32 The simple (version of the) **base_up_beh_0** behaviour

33 obtains the base triplet, **b_tri**, and the base resumption, **b_qui**,

34 performs the **base_up_fct** planning function and

35 provides its result, a resumption, **b_qui′**,
to the quintuplet repository,

36 whereupon it reverts to being **base_up_beh_0**.

value

```
32  base_up_beh_0() ≡
33      let (b_tri,b_qui) = (b_tri_ch?,b_qui_ch?) in
34      let b_qui′ = base_up_fct(b_tri)(b_qui) in
35      b_qui_ch ! b_qui′ end end ;
36      base_up_beh_0()
```

- The base_up_beh_0 behaviour
  - repeatedly "performs" urban planning, "from scratch",
  - as if new urban space, auxiliary and requirements information was "new" in every re-planning
  - — "ad infinitum" !
- We now revise base_up_beh_0 into base_up_beh_1
  - — a behaviour "almost" like base_up_beh_0,
  - but one which may terminate.

37 base_up_beh_1

38 first behaves like base_up_beh_0 (Items 33–35)

39 then checks whether the obtained base resumption is satisfactory, that is, is OK as an end-result of base urban planning.

40 If so then base_up_beh_1 terminates,

41 else it resumes being base_up_beh_1.

value

```
31    base_up_beh_1: Unit → in b_tri_ch out b_qui_ch Unit
37    base_up_beh_1() ≡
38        let b_qui = b_base_up_fct(b_tri_ch?)(b_qui_ch?) in b_qui_ch!b_qui ;
39        if b_qui_satisfactory(b_qui)
40            then skip
41            else base_up_beh_1() end
37        end

39    b_qui_satisfactory: bQUI → Bool
```

- The b_qui_satisfactory predicate
  - inquires the base quintuplet, b_qui, as for its suitability as a final candidate for an urban plan[10].

[10]The b_qui_satisfactory argument, b_qui, embodies not only that plan, but also the basis for its determination.

3.7. That was all for today !

- Tomorrow I will cover a generalisation of urban planning.