Simple Pipeline Systems
A Domain Description, Endurants

Dines Bjørner
Fredsvej 11, DK-2840 Holte, Denmark
DTU Compute, DK-2800 Kgs. Lyngby, Denmark
E-mail: bjorner@gmail.com, URL: www.imm.dtu.dk/~dibj

January 15, 2017, 15:35

Abstract

We give an example of the description of the endurants of simple pipeline systems. The example was first developed during an MSc/PhD course in the fall of 2008 at the Technical University of Graz, Austria. The present decription is a reformulation of the 2008 description and is based on [5] (2014).

Contents

1 Some Informal Hints at 'Pipelines'

To direct the readers' attention to the example of a pipeline system we show some informal graphics and some photos.

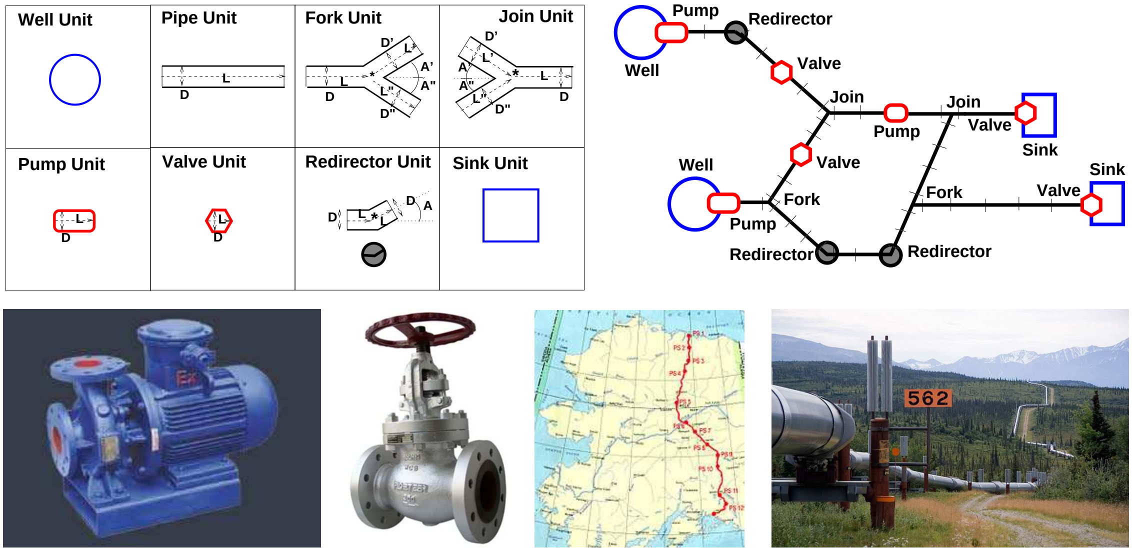

Figure 1: Oil unit graphics; a simple oil pipeline.
A pump; a valve; the Trans-Alaska Pipeline System (TAPS); TAPS pipes, redirectors and 'heat pipes'.

2 Parts

1 Our domain, Δ, consists of a pipeline, PL[1].

2 A pipeline consists of a set, US, of pipeline units[2].

3 A pipeline unit is either a well, or a pipe, or a pump, or a valve, or a fork, or a join, or a redirector[3], or a sink.[4]

4 All these unit sorts are atomic and disjoint[5].

type
1 Δ, PL
2 US = U-**set**
3 We, Pi, Pu, Va, Fo, Jo, Di, Si
3 U == We | Pi | Pu | Va | Fo | Jo | Rd | Si
value
1 **obs_part**_PL: Δ → PL
2 **obs_part**_US: PL → US
type
4 Well, Pipe, Pump, Valv, Fork, Join, Dir, Sink
4 We::Well, Pi::Pipe, Pu::Pump, Va::Valv, Fo:Fork, Jo::Join, Rd:Rdi, Si::Sink

3 Material

Applying `observe_material_sorts` to any unit u:U we obtain

[1] **Methodology:** That is: Δ is composite with just one sort: PL
[2] **Methodology:** Thus PL has a concrete type: U-**set**
[3] Redirector units serve to let pipelines "meander" their way in the landscape, and over hilly terrain.
[4] **Methodology:** The immediate subparts of the pipeline domain are all atomic parts.
[5] **Methodology:** The type constructor :: makes the We, Pi, Pu, Va, Fo, Jo, Di, Si sorts disjoint

© Dines Bjørner. 2017, Fredsvej 11, DK–2840 Holte, Denmark – January 15, 2017, 15:35

5 a type clause stating the material sort LoG for some further undefined liquid or gaseous material, and a material observer function signature.

type
5 LoG
value
5 **obs_mat_**LoG: U → LoG

`has_materials`(u) is a prerequisite for **obs_mat_**LoG(u).

4 Unique Identifiers

6 We focus only on the unique identifiers of pipeline units.

type
6. UI
value
6. **uid_**U → UI

5 Mereology

Pipeline units serve to conduct fluid or gaseous material. The flow of these occur in only one direction: from so-called input to so-called output.

7 Wells have exactly one connection to an output unit.

8 Pipes, pumpsm valves and redirectors have exactly one connection from an input unit and one connection to an output unit.

9 Forks have exactly one connection from an input unit and exactly two connections to distinct output units.

10 Joins have exactly two connections from distinct input units and one connection to an output unit.

11 Sinks have exactly one connection from an input unit.

12 Thus we model the mereology of a pipeline unit as a pair of disjoint sets of unique pipeline unit identifiers.

type
12 UM′=(UI-**set**×UI-**set**)
12 UM={|(iuis,ouis):UM′•iuis ∩ ouis={}|}
value
12 **obs_mereo_**U: UM
axiom [Well−formedness of Pipeline Systems, PLS (0)]
∀ pl:PL,u:U • u ∈ **obs_part_**Us(pl) ⇒
let (iuis,ouis)=**obs_mereo_**U(u) **in**
case (**card** iuis,**card** ouis) **of**
7 (0,1) → **is_**We(u),
8 (1,1) → **is_**Pi(u)∨**is_**Pu(u)∨**is_**Va(u)∨**is_**Rd(u),
9 (1,2) → **is_**Fo(u),
10 (2,1) → **is_**Jo(u),
11 (1,0) → **is_**Si(u), _ → **false**
end end

© Dines Bjørner 2017, Fredsvej 11, DK–2840 Holte, Denmark – January 15, 2017, 15:35

5.1 Shared Connectors

Two pipeline units, p_i with unique identifier π_i, and p_j with unique identifier π_j, that are connected, such that an outlet marked π_j of p_i "feeds into" inlet marked π_i of p_j, are said to share the connection (modeled by, e.g., $\{(\pi_i,\pi_j)\}$)

5.2 Routes

13 The observed pipeline units of a pipeline system define a number of routes (or pipelines):

(a) The null sequence , $\langle\rangle$, of no units is a route.

(b) Any one pipeline unit, u, of a pipeline system forms a route, $\langle u\rangle$, of length one.

(c) Let $r_i{}^\frown\langle u_i\rangle$ and $\langle u_j\rangle{}^\frown r_j$ be two routes of a pipeline system.

(d) Let ui_i and ui_j be the unique identifiers u_i, respectively u_j.

(e) If one of the output connectors of u_i is ui_j

(f) and one of the input connectors of u_j is ui_i,

(g) then $r_i{}^\frown\langle u_i,u_j\rangle{}^\frown r_j$ is a route of the pipeline system.

```
type
13.   R = U^ω
value
13   routes: Δ →~ R
13   routes(δ) ≡
13      let us = obs_part_US(obs_part_PL(δ)) in
13(a)    let rs = {⟨⟩}
13(b)              ∪ {⟨u⟩|u:U•u ∈ us} ∪
13(g)              ∪ {ri⌢⟨ui⟩⌢⟨uj⟩⌢rj
13(c)                 | ri⌢⟨ui⟩,⟨uj⟩⌢rj:R • {ri⌢⟨ui⟩,⟨uj⟩⌢rj}⊆rs
13(d),13(e)             ∧ uid_U(uj)∈ xtr_iUIs(ui)
13(d),13(f)             ∧ uid_U(ui)∈ xtr_oUIs(uj)} in
13      rs end end

   xtr_iUIs: U → UI-set, xtr_iUIs(u) ≡ let (iuis,_) = obs_mereo_U in iuis end
   xtr_oUIs: U → UI-set, xtr_oUIs(u) ≡ let (_,ouis) = obs_mereo_U in iuis end
```

5.3 Wellformed Routes

14 The observed pipeline units of a pipeline system forms a net subject to the following constraints:

(a) unit output connectors, if any, are connected to unit input connectors;

(b) unit input connectors, if any, are connected to unit output connectors;

(c) there are no cyclic routes;

(d) nets has all their connectors connected, that is, "starts" with wells

(e) and "ends" with sinks.

```
value
14. wf_Net: Δ → Bool
14. wf_Net(δ) ≡
14.    let us = obs_part_US(obs_part_PL(δ)) in
14.    ∀ u:U • u ∈ us ⇒ let (iuis,ouis) = obs_mereo_(u) in
14.       axiom 7.–11.
```

14(a). $\wedge$ $\forall$ ui:UI•ui $\in$ iuis $\Rightarrow$ $\exists$ u′:U•u′$\neq$u$\wedge$u′ $\in$ us$\wedge$**uid**_U(u′)=ui $\wedge$ ui $\in$ xtr_iUIs(u′)
14(b). $\wedge$ $\forall$ ui:UI•ui $\in$ ouis $\Rightarrow$ $\exists$ u′:U•u′$\neq$u$\wedge$u′ $\in$ us$\wedge$**uid**_U(u′)=ui $\wedge$ ui $\in$ xtr_oUIs(u′)
14(c). $\wedge$ $\forall$ r:R•r $\in$ routes(δ) $\Rightarrow$ $\sim\exists$ i,j:**Nat**•i$\neq$j$\wedge${i,j}$\in$ **inds** r$\wedge$r(i)=r(j)
14(d). $\wedge$ $\exists$ we:We • we $\in$ us $\wedge$ r(1) = mkWe(we)
14(e). $\wedge$ $\exists$ si:Si • si $\in$ us $\wedge$ r(**len** r) = mkSi(si)
13. **end end**

6 Attributes

6.1 Geometric Unit Attributes

15 Common static unit attributes are Diameters and Lengths.

16 Well units have one output “Diameter”; pipe, Valve, Pump and Redirector units have Diameter; and Sink units have one input “Diameter”.

17 Pipe, valve and pumps units have Length.

18 Fork units have one input Diameter, two output Diameters: iD, oD_1, oD_2, and Lengths from input to a fork center, and from that to the two outputs: iL, oL_1, oL_2.

19 Join units have the “reverse”: one output Diameter, two input Diameters: oD, iD_1, iD_2, and Lengths from the two inputs to a join center, and from that to the single output: iL_1, iL_2, oL.

20 Redirector units have Lengths from the input to a “center” (where the unit redirection can be said to be “centered”), and from that center to the output: iL, oL.

type
15. D, L
value
16. **attr**_D: (We|Pi|Va|Pu|Rd|Si) $\rightarrow$ D
17. **attr**_L: (Pi|Va|Pu) $\rightarrow$ L
18. **attr**_Ds: Fo $\rightarrow$ (D$\times$(D$\times$D))
18. **attr**_Ls: Fo $\rightarrow$ (L$\times$(L$\times$L))
19. **attr**_Ds: Jo $\rightarrow$ ((D$\times$D)$\times$D)
19. **attr**_Ls: Jo $\rightarrow$ ((L$\times$L)$\times$L)
20. **attr**_Ls: Rd $\rightarrow$ L$\times$L

We omit detailing the angles with which the two segments emanate from the input segment of fork, the two segments are incident upon the put segment of a join, and a redirector deviates the output segment from its input segment. The oil unit graphics of Fig. 1 hints at these angles.

6.2 Unit Action Attributes

21 Valve units are either 100% `open`, or 100% `closed`. [6]

22 Pump units are either `pumping`, or `not_pumping`.[7]

type
21. OC == ″open″ | ″closed″
22. PS == ″pumping″ | ″not_pumping″
value
21. **attr**_OC: Va $\rightarrow$ OC
22. **attr**_PS: Pu $\rightarrow$ PS

[6]Without loss of generality we do not model fractional open/closed status.
[7]Without loss of generality we do not model fractional pumping status.

© Dines Bjørner 2017, Fredsvej 11, DK–2840 Holte, Denmark – January 15, 2017, 15:35

6.3 Spatial Unit Attributes

Pipelines are laid down in flat and hilly, even mountaineous terrain. Any one pipeline unit thus have spatial locations. We shall refrain from detailing (let alone formalising) the spatial attributes of units. But we can mention the following: Every unit has some spatial attributes: As material flow in units is one-directional we can associate with any unit a unique point, either their input or their output. Wells and sinks have their output, respectively their input being the unique such points. Forks have their input point. Joins have their output points, Pipe, valve, pump and redirection units have the single input from where the flow in that unit begins being their unique points. The non-unique points of a unit are now “at the ’opposite’ ends” of these units with respect to their unique points. Given that the unique points (inputs or outputs, as they may be) have a unique global position, the non-unique points “deviate” from those vertically and/or horizontally. Since we bring this (Appendix) example as an illustration of the use of analysis and description prompts, and not as an example of a full-fledged pipeline domain description, we shall refrain from systematically narrating and formalising these spatial unit attributes and the consequences of doing so[8].

6.4 Flow Attributes

We now wish to examine the flow of liquid (or gaseous) material in pipeline units. So we postulate a unit attribute Flow. We use two types

23 **type** Flow, F = Flow, L = Flow.

Productive flow, F, and wasteful leak, L, is measured, for example, in terms of volume of material per second. We then postulate the following unit attributes “measured” at the point of in- or out-flow or in the interior of a unit.

24 current flow of material into a unit input connector,
25 maximum flow of material into a unit input connector while maintaining laminar flow,
26 current flow of material out of a unit output connector,
27 maximum flow of material out of a unit output connector while maintaining laminar flow,
28 current leak of material at a unit input connector,
29 maximum guaranteed leak of material at a unit input connector,
30 current leak of material at a unit input connector,
31 maximum guaranteed leak of material at a unit input connector,
32 current leak of material from “within” a unit, and
33 maximum guaranteed leak of material from “within” a unit.

```
type
23 Flow, F = Flow, L = Flow
value
24 attr_cur_iF: U → UI → F
25 attr_max_iF: U → UI → F
26 attr_cur_oF: U → UI → F
27 attr_max_oF: U → UI → F
28 attr_cur_iL: U → UI → L
29 attr_max_iL: U → UI → L
30 attr_cur_oL: U → UI → L
31 attr_max_oL: U → UI → L
32 attr_cur_L: U → L
33 attr_max_L: U → L
```

The maximum flow attributes are static attributes and are typically provided by the manufacturer as indicators of flows below which laminar flow can be expected. The current flow attributes may be considered either reactive or biddable attributes.

It may be difficult or costly, or both, to ascertain flows and leaks in materials-based domains. But one can certainly speak of these concepts. This casts new light on domain modeling. That is in contrast to incorporating such notions of flows and leaks in requirements modeling where one has to show implementability. Modeling flows and leaks is important to the modeling of materials-based domains.

[8] The ‘consequences’ alluded to are those of the spatial well-formedness of pipelines.

34 For every unit of a pipeline system, except the well and the sink units, the following law apply.

35 The flows into a unit equal

(a) the leak at the inputs

(b) plus the leak within the unit

(c) plus the flows out of the unit

(d) plus the leaks at the outputs.

```
axiom [Well−formedness of Pipeline Systems, PLS (1)]
34    ∀ pls:PLS,b:B\We\Si,u:U •
34       b ∈ obs_part_Bs(pls)∧u=obs_part_U(b)⇒
34       let (iuis,ouis) = obs_mereo_U(u) in
35       sum_cur_iF(u)(iuis) =
35(a)         sum_cur_iL(u)(iuis)
35(b)       ⊕ attr_cur_L(u)
35(c)       ⊕ sum_cur_oF(u)(ouis)
35(d)       ⊕ sum_cur_oL(u)(ouis)
34       end
```

6.4.1 Intra Unit Flow and Leak Law

36 The sum_cur_iF (cf. Item 35) sums current input flows over all input connectors.

37 The sum_cur_iL (cf. Item 35(a)) sums current input leaks over all input connectors.

38 The sum_cur_oF (cf. Item 35(c)) sums current output flows over all output connectors.

39 The sum_cur_oL (cf. Item 35(d)) sums current output leaks over all output connectors.

```
36   sum_cur_iF:  U → UI-set →F
36   sum_cur_iF(u)(iuis) ≡ ⊕ {attr_cur_iF(u)(ui)|ui:UI•ui ∈ iuis}
37   sum_cur_iL: U → UI-set → L
37   sum_cur_iL(u)(iuis) ≡ ⊕ {attr_cur_iL(u)(ui)|ui:UI•ui ∈ iuis}
38   sum_cur_oF: U → UI-set → F
38   sum_cur_oF(u)(ouis) ≡ ⊕ {attr_cur_iF(u)(ui)|ui:UI•ui ∈ ouis}
39   sum_cur_oL: U → UI-set → L
39   sum_cur_oL(u)(ouis) ≡ ⊕ {attr_cur_iL(u)(ui)|ui:UI•ui ∈ ouis}
     ⊕: (F|L) × (F|L) → F
```

where ⊕ is both an infix and a distributed-fix function which adds flows and or leaks □

6.4.2 Inter Unit Flow and Leak Law

40 For every pair of connected units of a pipeline system the following law apply:

(a) the flow out of a unit directed at another unit minus the leak at that output connector

(b) equals the flow into that other unit at the connector from the given unit plus the leak at that connector.

```
     axiom [Well−formedness of Pipeline Systems, PLS (2)]
40       ∀ pls:PLS,b,b′:B,u,u′:U•
40          {b,b′}⊆obs_part_Bs(pls)∧b≠b′∧u′=obs_part_U(b′)
40          ∧ let (iuis,ouis)=obs_mereo_U(u),(iuis′,ouis′)=obs_mereo_U(u′),
40                ui=uid_U(u),ui′=uid_U(u′) in
40             ui ∈ iuis ∧ ui′ ∈ ouis′ ⇒
40(a)             attr_cur_oF(u′)(ui′) − attr_leak_oF(u′)(ui′)
40(b)           = attr_cur_iF(u)(ui) + attr_leak_iF(u)(ui)
40             end
40     comment: b′ precedes b □
```

From the above two laws one can prove the **theorem:** what is pumped from the wells equals what is leaked from the systems plus what is output to the sinks.

© Dines Bjørner 2017, Fredsvej 11, DK–2840 Holte, Denmark – January 15, 2017, 15:35

7 Bibliography

7.1 Bibliographical Notes

This paper shall be seen in the context of the following other papers:

- [5, Manifest Domains: Analysis & Description] lays the foundation for analysing & describing a large class of domains. It introduces two calculi; one of analysis and one of description prompts.
- [2, Domain Facets: Analysis & Description] continues that of [5] by enlarging the scope of phenomena being analysed and described.
- [1, Formal Models of Processes and Prompts] presents an operational semantics for the analysis and description processes and prompts covered in [5].
- [6, To Every Manifest Domain a CSP Expression] ***
- [4, From Domain Descriptions to Requirements Prescriptions] shows how to systematically 'derive' core elements of requirements from domain descriptions.
- [3, Domains: Their Simulation, Monitoring and Control] discusses software product lines of demos, simulators, monitors and monitors & controllers as they relate to descriptions and prescriptions for the product line domain.

Together these papers, the present and those referenced above, form a scientific and engineering basis for domain engineering.

7.2 References

[1] Dines Bjørner. Domain Analysis and Description – Formal Models of Processes and Prompts. *Submitted for consideration to Formal Aspects of Computing*, 2016. http://www.imm.dtu.dk-/~dibj/2016/process/process-p.pdf.

[2] Dines Bjørner. Domain Facets: Analysis & Description. *Submitted for consideration to Formal Aspects of Computing*, 2016. http://www.imm.dtu.dk/~dibj/2016/facets/faoc-facets.pdf.

[3] Dines Bjørner. Domains: Their Simulation, Monitoring and Control – A Divertimento of Ideas and Suggestions. Experimental Research, Fredsvej 11, DK–2840 Holte, Denmark, 2016. http://www.-imm.dtu.dk/~dibj/2016/demos/faoc-demo.pdf.

[4] Dines Bjørner. From Domain Descriptions to Requirements Prescriptions – A Different Approach to Requirements Engineering. *Submitted for consideration to Formal Aspects of Computing*, 2016.

[5] Dines Bjørner. Manifest Domains: Analysis & Description. *Formal Aspects of Computing*, ...(...):1–51, 2016. DOI 10.1007/s00165-016-0385-z http://link.springer.com/article/10.1007/s00165-016-0385-z.

[6] Dines Bjørner. To Every Manifest Domain a CSP Expression — A Rôle for Mereology in Computer Science. Submitted for consideration to Journal of Logical and Algebraic Methods in Programming, Fredsvej 11, DK–2840 Holte, Denmark, December 2016. http://www.imm.dtu.dk/~dibj/2016/mereo/mereo.pdf.