Pipelines
A Domain Description

Dines Bjørner
DTU Informatics, Techn.Univ.of Denmark
bjorner@gmail.com, www.imm.dtu.dk/~dibj

January 21, 2013: 16:12

1

Abstract 2

We present a description of an abstracted domain of pipelines.

The description structure follows that of the domain analysis structure into a description of endurant entities: the observation of parts and materials, the identification of unique part identifies, the mereology of parts and the (multitude) of part and material atttributes; and the description of perdurant entities: actions, evemts and behaviours. 3

In the present version (January 21, 2013) of this document we cover discrete and continuous endurants and discrete perdurants; but we do not cover continuous perdurants (actions, events, behaviours). The latter would require, as an example, description of continuous (laminar as well as turbulent) flows using, for example, Bernoulli and navier-Stokes (partial) differential equations. We are presently working on that problem.

Contents

1 Introduction 4

For illustrations of pipeline phenomena please refer to Appendix A.

2 Endurants 5

2.1 Parts

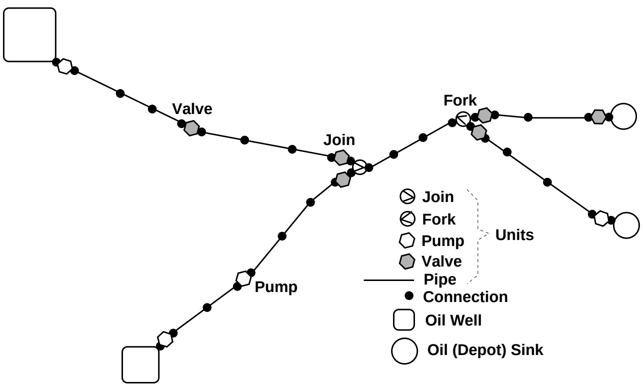

Figure 1: An oil pipeline system

6

1. A pipeline system contains a set of pipeline units and a pipeline system monitor.
2. The well-formedness of a pipeline system depends on its mereology (cf. Sect. 2.2.3) and the routing of its pipes (cf. Sect. 2.3.2).
3. A pipeline unit is either a well, a pipe, a pump, a valve, a fork, a join, or a sink unit.
4. We consider all these units to be distinguishable, i.e., the set of wells, the set pipe, etc., the set of sinks, to be disjoint.

7

type
1. PLS′, U, M
2. PLS = {| pls:PLS′•wf_PLS(pls) |}

value
2. wf_PLS: PLS → **Bool**

2. wf_PLS(pls) ≡ wf_Mereology(pls) ∧ wf_Routes(pls)
1. obs_Us: PLS → U-**set**
1. obs_M: PLS → M

type

3. U = We | Pi | Pu | Va | Fo | Jo | Si
4. We :: Well
4. Pi :: Pipe
4. Va :: Valv
4. Fo :: Fork
4. Jo :: Join
4. Si :: Sink

2.2 Part Identification and Mereology 8

2.2.1 Unique Identification

5. Each pipeline unit is uniquely distinguished by its unique unit identifier.

type

5. UI

value

5. uid_UI: U → UI

axiom

5. ∀ pls:PLS,u,u′:U•{u,u′}⊆obs_Us(pls)⇒u≠u′⇒uid_UI(u)≠uid_UI(u′)

2.2.2 Unique Identifiers 9

6. From a pipeline system one can observe the set of all unique unit identifiers.

value

6. xtr_UIs: PLS → UI-**set**
6. xtr_UIs(pls) ≡ {uid_UI(u)|u:U•u ∈ obs_Us(pls)}

7. We can prove that the number of unique unit identifiers of a pipeline system equals that of the units of that system.

theorem:

7. ∀ pls:PLS•**card** obs_Us(pl)=**card** xtr_UIs(pls)

2.2.3 Mereology 10

8. Each unit is connected to zero, one or two other existing input units and zero, one or two other existing output units as follows:

   a A well unit is connected to exactly one output unit (and, hence, has no "input").

b A pipe unit is connected to exactly one input unit and one output unit.

c A pump unit is connected to exactly one input unit and one output unit.

d A valve is connected to exactly one input unit and one output unit.

e A fork is connected to exactly one input unit and two distinct output units.

f A join is connected to exactly two distinct input units and one output unit.

g A sink is connected to exactly one input unit (and, hence, has no "output").

11

```
    type
8.    MER = UI-set × UI-set
    value
8.    mereo_U: U → MER
    axiom
8.     wf_Mereology: PLS → Bool
8.     wf_Mereology(pls) ≡
8.       ∀ u:U•u ∈ obs_Us(pls)⇒
8.         let (iuis,ouis) = mereo_U(u) in iuis ∪ ouis ⊆ xtr_UIs(pls) ∧
8.           case (u,(card uius,card ouis)) of
8a.              (mk_We(we),(0,1)) → true,
8b.              (mk_Pi(pi),(1,1)) → true,
8c.              (mk_Pu(pu),(1,1)) → true,
8d.              (mk_Va(va),(1,1)) → true,
8e.              (mk_Fo(fo),(1,1)) → true,
8f.              (mk_Jo(jo),(1,1)) → true,
8g.              (mk_Si(si),(1,1)) → true,
8.               _ → false end end
```

2.3 Part Concepts, I 12

2.3.1 Pipe Routes

9. A route (of a pipeline system) is a sequence of connected units (of the pipeline system).

10. A route descriptor is a sequence of unit identifiers and the connected units of a route (of a pipeline system).

```
    type
9.    R′ = U^ω
9.    R = {| r:Route′•wf_Route(r) |}
10.   RD = UI^ω
    axiom
10.   ∀ rd:RD • ∃ r:R•rd=descriptor(r)
    value
10.   descriptor: R → RD
10.   descriptor(r) ≡ ⟨uid_UI(r[i])|i:Nat•1≤i≤len r⟩
```

13

11. Two units are adjacent if the output unit identifiers of one shares a unique unit identifier with the input identifiers of the other.

```
    value
11.   adjacent: U × U → Bool
11.   adjacent(u,u′) ≡ let (,ouis)=mereo_U(u),(iuis,)=mereo_U(u′) in ouis ∩ iuis ≠ {} end
```

14

12. Given a pipeline system, pls, one can identify the (possibly infinite) set of (possibly infinite) routes of that pipeline system.
    a The empty sequence, $\langle\rangle$, is a route of pls.
    b Let u, u' be any units of pls, such that an output unit identifier of u is the same as an input unit identifier of u' then $\langle u, u'\rangle$ is a route of pls.
    c If r and r' are routes of pls such that the last element of r is the same as the first element of r', then $r\widehat{\ }\mathbf{tl}r'$ is a route of pls.
    d No sequence of units is a route unless it follows from a finite (or an infinite) number of applications of the basis and induction clauses of Items 12a–12c.

```
    value
12.   Routes: PLS → RD-infset
12.   Routes(pls) ≡
12a.     let rs = ⟨⟩ ∪
12b.           {⟨uid_UI(u),uid_UI(u′)⟩|u,u′:U•{u,u′}⊆obs_Us(pls) ∧ adjacent(u,u′)}
12c.         ∪ {r⌢tl r′|r,r′:R•{r,r′}⊆rs}
12d.     in rs end
```

2.3.2 Wellformed Routes 15

13. A route is acyclic if no two route positions reveal the same unique unit identifier.

```
    value
13. acyclic_Route: R → Bool
13. acyclic_Route(r) ≡ ∼∃ i,j:Nat•{i,j}⊆inds r ∧ i≠j ∧ r[i]=r[j]
```

16

14. A pipeline system is well-formed if none of its routes are circular (and all of its routes embedded in well-to-sink routes).

```
    value
14. wf_Routes: PLS → Bool
14. wf_Routes(pls) ≡
14.    non_circular(pls) ∧ are_embedded_in_well_to_sink_Routes(pls)

14. non_circular_PLS: PLS → Bool
14. non_circular_PLS(pls) ≡
14.    ∀ r:R•r ∈ routes(p)∧acyclic_Route(r)
```


17

15. We define well-formedness in terms of well-to-sink routes, i.e., routes which start with a well unit and end with a sink unit.

```
    value
15.  well_to_sink_Routes: PLS → R-set
15.  well_to_sink_Routes(pls) ≡
15.    let rs = Routes(pls) in
15.    {r|r:R•r ∈ rs ∧ is_We(r[1]) ∧ is_Si(r[len r])} end
```

18

16. A pipeline system is well-formed if all of its routes are embedded in well-to-sink routes.

```
16.  are_embedded_in_well_to_sink_Routes: PLS → Bool
16.  are_embedded_in_well_to_sink_Routes(pls) ≡
16.     let wsrs = well_to_sink_Routes(pls) in
16.     ∀ r:R • r ∈ Routes(pls) ⇒
16.        ∃ r′:R,i,j:Nat •
16.          r′ ∈ wsrs
16.        ∧ {i,j}⊆inds r′∧i≤j
16.        ∧ r = ⟨r′[k]|k:Nat•i≤k≤j⟩ end
```

2.3.3 Embedded Routes 19

17. For every route we can define the set of all its embedded routes.

```
    value
17. embedded_Routes: R → R-set
17. embedded_Routes(r) ≡
17.    {⟨r[k]|k:Nat•i≤k≤j⟩ | i,j:Nat• i {i,j}⊆inds(r) ∧ i≤j}
```

2.3.4 A Theorem 20

18. The following theorem is conjectured:

    a the set of all routes (of the pipeline system)

    b is the set of all well-to-sink routes (of a pipeline system) and

    c all their embedded routes

```
    theorem:
18.  ∀ pls:PLS •
18.  let rs = Routes(pls),
18.     wsrs = well_to_sink_Routes(pls) in
18a.  rs =
18b.      wsrs ∪
18c.      ∪ {{r′|r′:R • r′ ∈ embedded_Routes(r″)} | r″:R • r″ ∈ wsrs}
17.  end
```

2.4 Materials 21

19. The only material of concern to pipelines is the gas[1] or liquid[2] which the pipes transport[3].

type
19. GoL
value
19. obs_GoL: U → GoL

2.5 Attributes 22

2.5.1 Part Attributes

20. These are some attribute types:
    a estimated current well capacity (barrels of oil, etc.),
    b pipe length,
    c current pump height,
    d current valve open/close status and
    e flow (e.g., volume/second).

23

type
20a. WellCap
20b. LEN
20c. Height
20d. ValSta == open | close
20e. Flow

21. Flows can be added (also distributively) and subtracted, and
22. flows can be compared.

value
21. $\oplus,\ominus$: Flow×Flow → Flow
21. $\oplus$: Flow-**set** → Flow
22. $<,\leq,=,\neq,\geq,>$: Flow × Flow → **Bool**

24

23. Properties of pipeline units include
    a estimated current well capacity (barrels of oil, etc.),
    b pipe length,

[1] Gaseous materials include: air, gas, etc.
[2] Liquid materials include water, oil, etc.
[3] The description of this document is relevant only to gas or oil pipelines.

c current pump height,

d current valve open/close status,

e current $\mathcal{L}$aminar in-flow at unit input,

f current $\mathcal{L}$aminar in-flow leak at unit input,

g maximum $\mathcal{L}$aminar guaranteed in-flow leak at unit input,

h current $\mathcal{L}$aminar leak unit interior,

i current $\mathcal{L}$aminar flow in unit interior,

j maximum $\mathcal{L}$aminar guaranteed flow in unit interior,

k current $\mathcal{L}$aminar out-flow at unit output,

l current $\mathcal{L}$aminar out-flow leak at unit output,

m maximum guaranteed $\mathcal{L}$aminar out-flow leak at unit output.

25

value

23a. attr_WellCap: We → WellCap
23b. attr_LEN: Pi → LEN
23c. attr_Height: Pu → Height
23d. attr_ValSta: Va → VaSta
23e. attr_In_Flow$_\mathcal{L}$: U → UI → Flow
23f. attr_In_Leak$_\mathcal{L}$: U → UI → Flow
23g. attr_Max_In_Leak$_\mathcal{L}$: U → UI → Flow
23h. attr_body_Flow$_\mathcal{L}$: U → Flow
23i. attr_body_Leak$_\mathcal{L}$: U → Flow
23j. attr_Max_Flow$_\mathcal{L}$: U → Flow
23k. attr_Out_Flow$_\mathcal{L}$: U → UI → Flow
23l. attr_Out_Leak$_\mathcal{L}$: U → UI → Flow
23m. attr_Max_Out_Leak$_\mathcal{L}$: U → UI → Flow

2.5.2 Flow Laws, I 26

24. "What flows in, flows out !". For $\mathcal{L}$aminar flows: for any non-well and non-sink unit the sums of input leaks and in-flows equals the sums of unit and output leaks and out-flows.

Law:

24. ∀ u:U\We\Si •
24. sum_in_leaks(u) ⊕ sum_in_flows(u) =
24. attr_body_Leak$_\mathcal{L}$(u) ⊕
24. sum_out_leaks(u) ⊕ sum_out_flows(u)

27

value

sum_in_leaks: U → Flow
sum_in_leaks(u) ≡
let (iuis,) = mereo_U(u) **in**

```
        ⊕ {attr_In_Leak_ℒ(u)(ui)|ui:UI•ui ∈ iuis} end
    sum_in_flows: U → Flow
    sum_in_flows(u) ≡
        let (iuis,) = mereo_U(u) in
        ⊕ {attr_In_Flow_ℒ(u)(ui)|ui:UI•ui ∈ iuis} end
    sum_out_leaks: U → Flow
    sum_out_leaks(u) ≡
        let (,ouis) = mereo_U(u) in
        ⊕ {attr_Out_Leak_ℒ(u)(ui)|ui:UI•ui ∈ ouis} end
    sum_out_flows: U → Flow
    sum_out_flows(u) ≡
        let (,ouis) = mereo_U(u) in
        ⊕ {attr_Out_Leak_ℒ(u)(ui)|ui:UI•ui ∈ ouis} end
```

28

25. "What flows out, flows in !". For ℒaminar flows: for any adjacent pairs of units the output flow at one unit connection equals the sum of adjacent unit leak and in-flow at that connection.

Law:

```
25. ∀ u,u′:U•adjacent(u,u′) ⇒
25.     let (,ouis)=mereo_U(u), (iuis′,)=mereo_U(u′) in
25.     assert: uid_U(u′) ∈ ouis ∧ uid_U(u) ∈ iuis ′
25.     attr_Out_Flow_ℒ(u)(uid_U(u′)) =
25.     attr_In_Leak_ℒ(u)(uid_U(u))⊕attr_In_Flow_ℒ(u′)(uid_U(u)) end
```

2.5.3 Material Attributes 29

3 Perdurants 30

4 Conclusion 31

A Illustrations of Pipeline Phenomena 32

A.1 Photos of Pipeline Units and Diagrams of Pipeline Systems

A.1.1 Pipeline Nets

33

A.1.2 Pipes 34

A.1.3 Valves 35

A.1.4 Pumps 36

A.1.5 Compressors 37

A.1.6 Pigs 38

39

Figure 2: The Planned Nabucco Pipeline: http://en.wikipedia.org/wiki/Nabucco_Pipeline

Figure 3: The Planned Nabucco Pipeline: http://en.wikipedia.org/wiki/Nabucco_Pipeline

Figure 4: Pipes

Figure 5: Valves

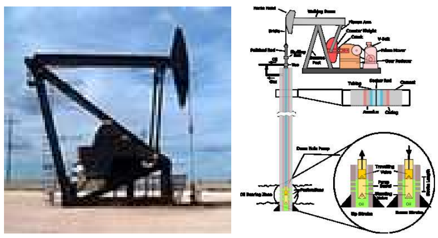

Figure 6: Oil Pumps

Figure 7: Gas Compressors

Figure 8: New and Old Pigs

Figure 9: Pig Launcher, Receiver