Railways
A Domain Model

Dines Bjørner
DTU Compute, Technical University of Denmark
Fredsvej 11, DK-2840 Holte, Danmark
E–Mail: bjorner@gmail.com, URL: www.imm.dtu.dk/~db

September 30, 2025

I AM TIRED. MUST, SORRY, GIVE UP FURTHER WORK ON THIS PAPER. 30.9.2025.

Abstract

I, with colleagues and students, have modeled various aspects of railways since 1993! Yet, I try again! As much for amusement as for pastime. Well, also for collecting my thoughts!

The **first** main domain modeling efforts – for segments of the railway domain – were carried out by Søren Prehn, Chris W. George and me at UNU/IIST, the *UN University's International Institute for Software Technology* Macau, 1991–1997. But not approaching the modeling as prescribed in my *Domain Science & Engineering* approach – such as I shall be doing it here.

So what more fitting, to, in a sense, "end" my 30 or so year quest, by returning to the first domain: railways!

Contents

1 Introduction

Since my days as Director of the *UN University's International Institute for Software Technology, UN/IIST*, Macau, 1991–1997, I have been modeling various aspects of railway systems. It began in 1992 with the Ministry of Railways of China: 4-5 UN Fellows came to UNU/IIST, where under the leadership of, first, Søren Prehn, then Chris W. George, 1992–1995, we analyzed & described a model of train traffic. From there requirements were prescribed and software [code] was designed and developed for the *running map* monitoring and control of train traffic along rail lines [22, 29, 28, 19, 20, 21].

From this sprung a series of studies: [5, 6, 25, 7, 27, 26, 2, 3, 24, 1, 4, 23]

Some of these papers and reports we collected, by me, AlbenaS Strupchanska and Martin Pěnička: *Railways*, 1995–2004, Techn. Univ. of Denmark - a compendium `www.imm.dtu.dk/~dibj/train-book.pdf`,

All of these studies were done well before the specific approach to domain modeling that I have advocated emerged: [8, 9, 10, 11, 13, 14, 15, 16, 17, 18].

So, the model presented in this report follows the specific domain analysis & description approach advocated since 2016 [11, 13, 14, 15, 16, 17, 18].

The reader may be served well by "studying" carefully the above table-of-contents: assuring that we present the analysis & description in the "standard way" of doing so!

As of September 30, 2025 I have neither checked the narratives and their normalization's, nor endeavored to achieve a most elegant formulation (incl., formalization) of all attribute and function definitions. I, so I admit, have, in a sense, rushed to complete individual [sub]paragraphs. In a proper domain model development, by a team of – in this example case, 4–5 professional domain modelers – each project day would start with individual project members reviewing a colleagues' work, on a rotation-shift base; followed by a project group coffee/tea mid-morning meeting where also issues of elegance are reviewed. The aim of these development steps is to ensure highest quality of work.

2 The Model

2.1 The Universe of Discourse

2.1.1 Sketch "Narration"

1. Let the **universe of discourse** be identified by the short name RS – for railway system. We "sketch-identify" the following endurants [parts]:

   (a) **train**s –

   (b) as composed from [train] **car**s,

   and a railway net composed from:

   (c) **line**s – "uninterrupted stretches or rail,

   (d) **junction**s – where lines "meet",

   (e) **station**s – as "special" aggregates of lines, junctions, platforms and sidings,

   and the reason for having trains:

   (f) **passenger**s,

   (g) **goods**.

   And we sketch identify the following perdurants [behaviours]:

   (a) **train**[1]

   (b) **line**,

   (c) **junction**,

   (d) **station**,

   (e) **platform**,

   (f) **siding**, and

   (g) **passenger**

This narrative suggests a rather different "decomposition" than found in [5, 6, 25, 7, 27, 26, 2, 3, 24, 1, 4, 23].

We omit "sketch-identifying" endurant identification, mereology and attributes.

2.1.2 Sketch "Formalization"

type
1. RS
1a. Train
1b. Car
1c. Line
1d. Junction
1e. Station
1f. Passenger
1g. Good
value
1a. train
1b. line
1c. junction

[1] Disregard, for the moment, the seeming "overloading" of names.

1d. station

1g. passenger

What is it that we wish to "capture" in our domain model?

It is the movement of trains, in and out of stations, along lines, through junctions; it is the embarking and disembarking of passengers and the loading and unloading of goods – at station platforms, respectively station sidings; it is the ticketing of passengers for simple or composite journeys.

2.2 The Endurants

2.2.1 External Qualities

The Main Endurants – i.e., Parts

2. The railway system is
   (a) an aggregate of a rail net,
   (b) an aggregate [set] of trains,
   (c) an aggregate [set] of passengers, and
   (d) an aggregate [set] of goods.

   A rail net is an aggregate of
   (a) [sets of] lines,
   (b) [sets of] junctions
   (c) and [sets of] stations.
3. Lines are presently further undefined.
4. Junctions are presently further undefined.
5. Stations are presently further undefined.

Lines, junctions, stations, passengers and goods are presently considered atomic.

type
2 . RS
2a. RN
2b. AT, TS = Train-**set**
2c. AP, PS = Passenger-**set**
2d. AG, GS = Goods-**set**
2a. AL, LS = L-**set**
2b. AJ, JS = J-**set**
2c. AS, SS = S-**set**
3. L
4. J
5. S
value
2a. **obs**_RN: RS → RN
2c. **obs**_AP: RS → AP, **obs**_PS: AP → PS
2d. **obs**_AG: RS → AG, **obs**_GS: AG → GS
2a. **obs**_AL: RN → AL, **obs**_AS: AL → LS
2b. **obs**_AJ: RN → AJ, **obs**_AJ: AJ → JS
2c. **obs**_AS: RN → AS, **obs**_SS: AS → SS
2b. **obs**_AT: RS → AT, **obs**_TS: AT → TS

A Railway State: Domain states are sets of [manifestable] parts.

6. We choose, here, to consider primarily the set of sets of atomic parts: lines, junctions, stations, passengers and goods,

7. but also trains (not their constituent cars).

8. The union of these form a state.

variable
6. rn:RN := **obs**_RN(rs)
6. at:AT := **obs**_AT(rs)
6. ls:L-**set** := **obs**_AS(**obs**_AL(rs))
6. js:J-**set** := **obs**_JS(**obs**_AJ(rs))
6. ss:S-**set** := **obs**_AS(**obs**_AS(rs))
6. ps:P-**set** := **obs**_AS(**obs**_AL(rs))
6. gs:G-**set** := **obs**_GS(**obs**_AG(rs))
7. ts:T-**set** := **obs**_TS(**obs**_AT(rs))
8. $\sigma := ls \cup js \cup ss \cup ps \cup gs \cup ts$

Comments: We have decided to model the composite rail net, the composite aggregate of trains, and all the atomic parts. The rail net part, we envision, will be used to model that there is one, say national, authority that monitors & controls the rail net, and that there may be more than one train company (with separate trains), and that there is there a, say national, authority that monitors & controls these rail companies: allocate *time* & *space* on the rail net ■

Train Parts:

MORE TO COME

Stations:

TO BE WRITTEN

2.2.2 Internal Qualities

Unique Identification

9. The rail net, the aggregate of trains, lines, junctions, stations, passengers, goods, and trains have unique identification.

10. The union of the rail net, the aggregate of trains, and all the sets of line, junction, station, passenger, good, and train identifiers for a state.

type
9. RNI, ATI, LI, JI, PI, GI, TI
value
9. **uid**_L: L→LI, **uid**_J: J→JI, **uid**_S: S→SI, **uid**_P: P→PI, **uid**_G: G→GI, **uid**_L: T→TI

variable
10. rn_{uid} := **uid_**(rn)
10. at_{uid} := **uid_**(at)
10. ls_{uid} := { **uid_**(l) | l:L • l ∈ ls }
10. js_{uid} := { **uid_**(j) | j:J • j ∈ js }
10. ss_{uid} := { **uid_**(s) | s:S • s ∈ ss }
10. ps_{uid} := { **uid_**(p) | p:P • p ∈ ps }
10. gs_{uid} := { **uid_**(t) | g:G • g ∈ gs }
10. ts_{uid} := { **uid_**(t) | t:T • t ∈ ts }
10. σ_{uid} := { rn_{uid},at_{uid}} ∪ ls_{uid} ∪ js_{uid} ∪ ss_{uid} ∪ ps_{uid} ∪ gs_{uid} ∪ ts_{uid}

11. The identifiers of parts are unique, that is: their number is equal to the number of state parts. Always (□) – over time, as the net may be expanded/reduced, trains, passengers and goods may be added or removed[2]

axiom [Unique Identification]
11. □ **card** σ = **card** σ_{uid}

Part Retrieve Functions

12. From a line, junction and station identifier we can retrieve the identified line, junction and station.

value
12. retr_L: LI → L
12. retr_L(li) ≡ l:L • l ∈ ls ∧ **uid**_L(l)=li
12. **pre** ∃ l:L • l ∈ ls
12. retr_J: JI → J
12. retr_J(ji) ≡ j:J • j ∈ ls ∧ **uid**_J(j)=ji
12. **pre** ∃ j:J • j ∈ js
12. retr_S: SI → S
12. retr_S(si) ≡ ! s:S • S ∈ ss ∧ **uid**_S(s)=si
12. **pre** s:S • s ∈ ss

Mereology To understand the role of lines, junctions and stations of a railway net we study Fig. 1 on the facing page. It depicts four stations: `Sa, Sb, Sc, Sd`, ten lines: Lx, Lo, Li, Lj, Lv, Lw, Lj, Lk, Ly, Lq and one junction `J`. Figure 1 on the next page only shows a simple example, in the dotted/dashed part of the figure, of a junction. Figure 1 on the facing page shows that the stations are connected by pairs of lines. That is not a mandate.

13. The mereology of the rail net is the set of all station, line and junction identifiers.

[2]We shall not model the dynamics of changes (additions, deletions) from the rail net, or of trains, passengers and goods in this report.

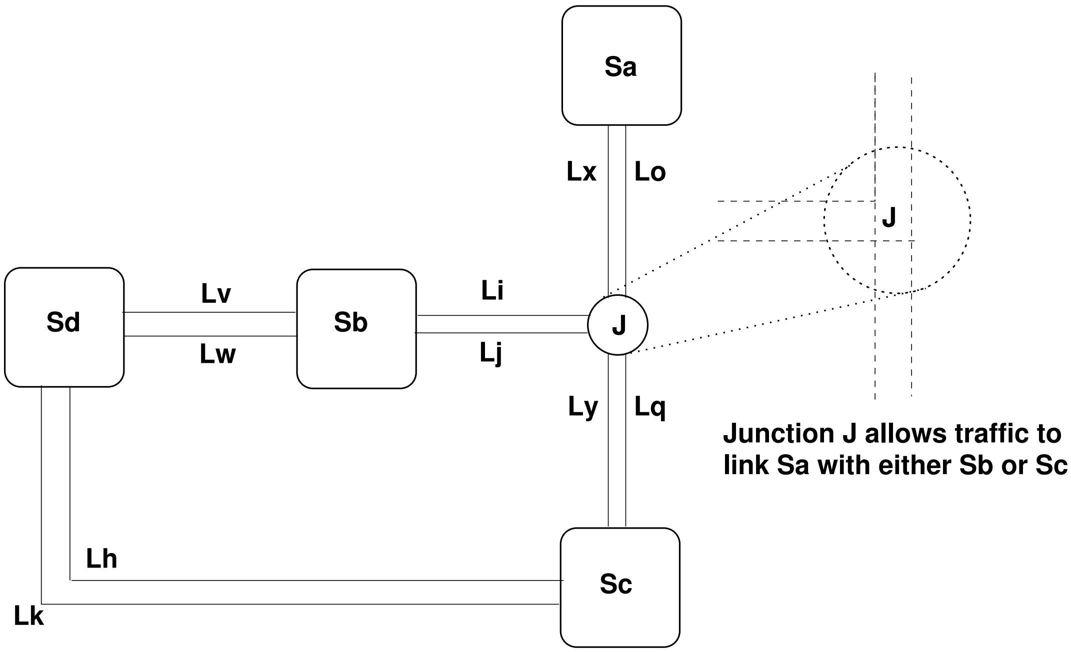

Figure 1: Fragment of a Rail Net

14. The mereology of the aggregate of trains is a set of disjoint sets of train identifiers –each set, in a way, designating the trains of a train company[3].

15. The mereology of a line is a pair: a two element set of identifiers of stations and junctions – that the lines is connected to – in the net, and the set of identifiers of the trains that may move along that line.

16. The mereology of a junction is a pair: a set of a two element set of identifiers of stations and lines identifiers – that the junction is connected to – in the net, and the set of identifiers of the trains that may move along that junction.

17. The mereology of a station is the two element set of identifiers of lines and junctions identifiers – that the station is connected to – in the net, and the set of identifiers of the trains that may move in and out of the station.

18. The mereology of a passenger is the set of identifiers of trains that the passenger may travel on – in the railway system.

19. The mereology of a [collection of] goods is the set of identifiers of trains that the goods may be transported with – in the railway system.

[3]`NSF, Union Pacific, CSX, Norfolk Southern, SNCF, Avanti West Coast, DB, SJ, NSB, DSB, Trenitalia, Renfe, Flixtrain, ÖBB, SBB, SNB, NS, ...`

20. The mereology of a train is the set of identifiers of lines, junctions and stations that it may "visit" – in the net!

type
13. MRN = (SI|LI|JI)**-set**
13. **axiom** ∀ mrn:MRN • mrn = ss_{uid} ∪ ls_{uid} ∪ js_{uid}
14. MAT = (TI**-set**)**-set**
14. **axiom** ∀ mrn:MRN • ∀ tis,tis′•{tis,tis′}⊆ ∧ (tis≠tis′)⇒(tis∩ti′={})
15. ML = ((JI|SI)**-set**×TI**-set**)
15. **axiom** ∀ (jsis,tis):ML : −**card** jsis =2 ∧ jsis ⊆ js_{uid}∪ss_{uid} ∧ tis ⊆ ts_{uid}
16. MJ = ((LI|SI)**-set**)**-set**×TI**-set**
16. **axiom** ∀ (lsis,tis):MJ • lsis ⊆ ls_{uid}∪ss_{uid} ∧ tis ⊆ ts_{uid}
17. MS = ((LI|JI)**-set**)**-set**
17. **axiom** ∀ ljis:(LI|JI)**-set** • ljis ∈ mj:MJ•**card** ljis =2 ∧ ljis ⊆ ls_{uid}∪js_{uid}
18. MP = TI**-set**, **axiom** ∀ mp:MP • mp ⊆ ps_{uid}
19. MG = TI**-set**, **axiom** ∀ mg:MG • mg ⊆ ts_{uid}
20. MT = (SI|JI|LI)**-set**
20. **axiom** ∀ mt:MT • mt ⊆ ss_{uid}∪js_{uid}∪ls_{uid}

value
13. **mereo**_RN: RN → MRN
14. **mereo**_AT: AT → MAT
15. **mereo**_L: L → ML
16. **mereo**_J: J → MJ
17. **mereo**_S: S → MS
18. **mereo**_P: P → MP
19. **mereo**_G: G → MG
20. **mereo**_T: T → MT

Routes: A rail net thus defines a concept of routes.

21. A route is a [finite] sequences of stations, lines and junctions identified by their unique identifiers.
22. Routes satisfy a wellformedness criterion:
    (a) A route is either empty, or
    (b) a non-empty route consists of at least a pair of station-line, or line-station, or line-junction, or junction-line identifiers that are connected:
        i. Let i and j be any pair of
        ii. adjacent identifiers in a route.
        iii. If of a station, respectively of a link, then they must be in one an-others mereology, that is: connected;
        iv. similar for other types of such [adjacent pairs of] identifiers.

type
21. R = (SI|LI|JI)*

21. $\Box\ \forall$ r:R • **elems** r $\subseteq \sigma_{uid}$
value
22. wf_R: R → **Bool**
22a. wf_R(r) ≡
22a. r = ⟨⟩
22b. ∨ ∀ (i,j):(SI×LI)|(LI×SI)|(LI×JI)|(JI×LI) •
22(b)i. r = lr⌢⟨i,j⟩⌢rr ∧
22(b)ii. **let** ei = rtr_E(i)(σ), ej = rtr_E(j)(σ) **end**
22(b)iii. is_S(ei) ∧ is_L(ej) ⇒ i ∈ **mereo**_L(ej) ∧ j ∈ **mereo**_S(ei)
22(b)iv. is_L(ei) ∧ is_S(ej) ⇒ i ∈ **mereo**_S(ej) ∧ j ∈ **mereo**_L(ei)
22(b)iv. is_L(ei) ∧ is_J(ej) ⇒ i ∈ **mereo**_L(ej) ∧ j ∈ **mereo**_J(ei)
22(b)iv. is_J(ei) ∧ is_L(ej) ⇒ i ∈ **mereo**_S(ej) ∧ j ∈ **mereo**_L(ei)

23. We define a function, `routes`, which
    (a) postulates a possibly infinite set, rs, of finite routes between any two stations, lines or junctions –
    (b) such that there are no well-formed routes, p, over the station, lines and junctions of the net which are not in rs.

23. routes: RN → R-**infset**
23a. routes(rn) ≡ { r | r:R ∧ wf_R(r) }

24. A `is_circular_route`, is a route which does not "list" the same [station, or more generally line and junction] identifier more than once.
25. A subset, `non_circular_station_routes`, of non-circular routes begin and end with a station identifier.

value
24. has_circular_routes: RS → **Bool**
24. has_circular_routes(rs) ≡ ∼∃ r:R • r ∈ routes(rs) ∧ is_circular_route(r)

24. is_circular_route: R → **Bool**
24. is_circular_route(r) ≡ ∃ i,j:**Nat** • {i,j}⊆**inds**(r) ∧ (i≠j) ∧ r[i]=r[j]

25. non_circular_station_routes: RS → R-**set**
25. non_circular_station_routes(rs) ≡
25. { r | r:R • r ∈ routes(rs) ∧ ∼is_circular_route(r) ∧ is_SI(r[1])∧is_SI(r([**len** r])) }

Attributes:

Lines[4]

[4] A line is a single rail track between a pair of either stations, or a station and a junction.

26. Lines have length. We “measure” lengths in terms of a natural number and meters![5]
27. Lines have states: the zero, one or two directions that trains may travel.
28. Lines have state spaces: a set of line states.
29. Lines have histories: sequences of time-stamped events of trains “passing by”.[6].

type
26. LLen = **Nat** × **meter** [STATIC]
27. LΣ = ((SI×(SI|JI)) | ((SI|LI)|SI) | (JI×(SI|LI)) | ((SI×LI)|JI))**-set** [PROGRAMMABLE]
28. LΩ = LΣ**-set** [STATIC]
29. LHist = (𝕋𝕀𝕄𝔼×LEvt[7])* [PROGRAMMABLE]
value
26. **attr**_LLen: L → LLen
27. **attr**_LΣ: L → LΣ
28. **attr**_LΩ: L → LΩ
29. **attr**_LHist: L → LHist

Junctions

30. Junctions have length[8]
31. Junctions have states: sets of pairs of the identifiers of lines connected to the junction[9].
32. Junctions have state spaces:
33. Junctions have histories: sequences of time-stamped awareness of trains “passing through” ![10]

type
30. JLen = NAT × **meter** [STATIC]
31. JΣ = (LI×LI)**-set** [PROGRAMMABLE]
32. JΩ = JΣ**-set** [STATIC]
33. JHist = (𝕋𝕀𝕄𝔼×JEvt)* [PROGRAMMABLE]

value
30. **attr**_JLen: J → JLen
31. **attr**_JΣ: J → JΣ
32. **attr**_JΩ: J → JΩ
33. **attr**_JHist: J → JHist

With length attributes of lines, junctions and stations we can now define notions of route lengths and of shortest routes.

34. The length of a[ny] non-circular route is the sum of the lengths of the lines, junctions and stations of the route.
35. To “calculate” route lengths we functions which “retrieve” stations, lines and junctions from station, line and junction identifiers – see Items 12 on page 6.

[5] We refer to [15, *Sect. 5.4.5*] and [12].
[6] Link events are such as `train entering` and `leaving link, progressing, at a link position`, and `train stopping along the link.`
[8] A junction length is a measure for the distance a train has to move from entering, from a line, a junction and, subsequently, leaving the junction and entering a line.
[9] If a pair: (li, lj) in in $l\sigma$ then trains can enter from line li and exit to line lj.
[10] Junction events are such as `train entering` and `leaving junction.`

```
type
34. Len = Nat × meter
value
34. i_to_i_length: R → Len
34.     case r of
34.         ⟨⟩ → 0,
34.         ⟨i⟩^r′ → retr_Len(i) + i_to_i_length(r′)
34.     end
34. pre:

value
35. retr_Len: (SI|LI|JI) → Len
35. retr_Len(i) ≡
35.    is_SI(i) → attr_SLen(retr_S(i)),
35.    is_LI(i) → attr_LLen(retr_L(i)),
35.    is_JI(i) → attr_JLen(retr_J(i))
```

36. Any two stations defines a set of zero, one or more routes "between" them.

37. And we can define a function which "selects" a shortest such route.

```
value
36. SSRoutes: SI × SI × RS → Routes
36. SSRoutes(fsi,tsi,rs) ≡
36.    let rts = routes(rs) in
36.    { r | r:R • r ∈ rts ∧ r[1]=fsi ∧ r[len r]=tsi } end
36.    fsi≠tsi

37. ShortestSSRoutes: SI × SI × RS → Routes
37. ShortestSSRoutes(fsi,tsi,rs) ≡
37.    let rts = SSRoutes(fsi,tsi,rs) in
37.    { r | r:R • r ∈ rts ∧ ∼∃ sr:R•sr ∈ rts ∧ i_to_i_length(r) < i_to_i_length(sr) } end
37.    fsi≠tsi
```

Stations

38. Stations have length![11]

39. Stations have one or more identified platforms

40. – where we do not define what a platform is!

41. Stations have states: a pair of sets of identifiers of lines connected to the station – indication from which lines trains may enter the station, respectively onto which lines trains may leave the station.

[11] A station length is a measure for the distance a train has to move from entering a station to be at a platform, respectively for leaving a platform and entering a line.

42. Stations have state spaces: sets of station states.

43. Stations have histories: sequences of time-stamped awareness of trains entering, stopping at, and leaving stations.

type
38. SLen = **Nat** × **meter** [STATIC]
39. SPlatforms = PId $\overrightarrow{m}$ Platform [STATIC]
40. Platform
41. SΣ = (LI-**set**×LI-**set**) **axiom** ∀ (flis,tlis):SΣ • flis ∪ tlis ⊆ ls_{uid}[PROGRAMMABLE]
42. SΩ = SΣ-**set** [STATIC]
43. SHist = (𝕋𝕀𝕄𝔼× SEvt)* [PROGRAMMABLE]
value
38. **attr**_SLEN: S → Len
38. **attr**_Platforms: S → Platforms
41. **attr**_SΣ: S → SΣ
42. **attr**_SΩ: S → SΩ
43. **attr**_SHist: S → SHist

Train Aggregate

The train aggregate has, as its – in the context of the present domain model – most relevant attributes:

44. a time-table that maps train identifiers to train identified journeys.

45. Train journeys maps time-intervals – in the form of begin/end times – into journeys – such that

46. for all such journeys

47. their time intervals

48. [are either, trivially, the same, or]

49. or are [mutually] non-overlapping.

50. A journey is a station arrival/departure annotated `adr:ADR`, non_circular_station_route –

51. here all station identifications are paired with arrival and departure times.[12]

type
44. TATT = TI $\overrightarrow{m}$ TJournies [PROGRAMMABLE]
45. TJournies = (𝕋𝕀𝕄𝔼×𝕋𝕀𝕄𝔼) $\overrightarrow{m}$ (JId × TJourney)
46. **axiom** ∀ tjs:TJournies •
47. ∀ (ft,tt),(ft′,tt′):**dom** tjs •

[12]The axiom formalization, next, were "rattled-of-the-top-of-my-head" on Sat. morning, Sept. 13, 2025 – **should be checked ! ! !.**

```
48.                     ((ft,tt)=(ft′,tt′))
49.                   ∨ ((ft<tt) ∧ (tt<ft′) ∧ (ft′<tt′))
51.          ∧ ∀ (jid,jr),(jid′,jr′) ∈ rng tjs ⇒
51.                 ((jid,jr)=(jid′,jr′)) ∨
51.                 ((jid≠jid′) ∧ (jr≠jr′))
51.               • jrs ∈ dom jrs • ∀ (jid,jr),(jid′,jr′)
51.          ∧ ∀ (,jr)(JId×Journey) • (_jt) ∈ rng tjs ⇒ stripped(jr) ∈ non_circular_station_routes(rs)
51.      TJourney = TADR
51.      TADR = ((𝕋𝕀𝕄𝔼×SI×𝕋𝕀𝕄𝔼)|LI|JI)*  axiom ∀ adr:ADR • ascending(adr)
value
51. ascending: ADR → Bool
51. ascending(adr) ≡ ...
51. stripped: ADR → R
51. stripped(adr)≡ ...
value
44. attr_TATT: AT → TATT
```

Trains

52. Trains are, at any one time, either not in service, or in service, acting as per a specific, identified, annotated journey.

    The annotations concern only the station element of a journey.

    Either the train stops or does not stop at the station.

    If it stops the annotation further states the planned arrival and departure times to and from the station.

    If it does no stop annotation [only] states the planned time of passing through the station.

53. Trains have a an expected speed.

54. The train arrival/departure times are commensurate with the planned train speeds and the line and junction length between stations.

55. Trains, in service, have current positions: on a line, between two junctions and/or stations, some fragment "down" the line from its "previous" rail net element, at a junction, or at a station.

56. Trains are either moving or are stopped. (speed = 0 **km/sec**).

57. Trains, when moving, have non-zero speed[13].

58. Trains carry sets of passengers and sets of goods.

59. Trains have histories: time-stamped sequences of its journey (station visits, line positions, junctions, passengers, goods).

[13]– we omit considerations of direction, i.e., velocity, and of acceleration

60. No two trains operate on that same journey.

type
52. TService = (Sta|LI|JI)* [PROGRAMMABLE]
52. Sta == Stop | NoStop
52. Stop :: s_si:SI×PID×𝕋𝕀𝕄𝔼×𝕋𝕀𝕄𝔼
52. NoStop :: s_si:SI×**"no_stop"**×𝕋𝕀𝕄𝔼
54. TSpeed = **Nat** × **km/hour** [STATIC]
value
52. wf_TService: TService → RTS → **Bool**
52. wf_TService(ts)(rts) ≡
52. **let** rt = conv_tservice_route(ts) **in** rt ∈ SSRoutes(rts) **end**
52.
52. conv_tservice_route: TService → Route
52. conv_tservice_route(ts) ≡
52. ι art:(SI|JI|LI)* • rt=**len** ts ∧
52. ∀ i:**Nat** • i ∈ **inds** ts • is_(LI|JI)(ts[i]) ⇒ art[i]=ts[i] ∧ rt[i]=si
type
55. TPos == OnL | atJ | inS [PROGRAMMABLE]
55. OnL :: (SI|JI) × F > (JI|SI)
55. F = **Real**, **axiom** ∀ f:F • 0<f<1
55. atJ :: JI
55. inS :: SI × PId
56. TMS == TMov | TSto [PROGRAMMABLE]
56. TMov :: Speed
1a. Speed = **Nat** × **km/hour**
58. TLoad == (PI|GI)-**set** [PROGRAMMABLE]
59. THist = (𝕋𝕀𝕄𝔼 × TEvt)* [PROGRAMMABLE]

value
60. wf_Traffic: RS → **Bool**
60. wf_Traffic(rs) ≡ ... [ ]

52. **attr**_Service: T → Service
55. **attr**_TPos: T → TPos
56. **attr**_TMS: T ⇒ TMS
58. **attr**_TLoad: T → TLoad
59. **attr**_THist: T → THist

Passengers

61. Passengers are either on a train or not.

62. Passenger have tickets – of their [most recent previous, present, or upcoming] journeys, zero or more –

63. that show some journey details: departure time, departure station, seat, arrival station, arrival time.

64. Passengers have histories: time-stamped sequences of events that the passenger has had on its train journeys, from a past into the present.

type
61. PState == [TI] [PROGRAMMABLE]
62. PTickets = Ticket* × [Ticket] × Ticket* [PROGRAMMABLE]
63. Ticket = $\mathbb{TIME}$ × SI × Seat × SI × $\mathbb{TIME}$
63. Seat = ...
64. PHist = ($\mathbb{TIME}$ × PEvt)* [PROGRAMMABLE]
64. PEvt = **boarding** | **disembarking** | ...
value
61. **attr**_PState: P → PState
62. **attr**_Tickets: P → Tickets
64. **attr**_PHist: P → PHist

The Rail Net Aggregate

With the rail net aggregate we can, for example, associate the, [again] for example national, operator that manages, monitors and controls the net: (i) assigns rights [for train companies] for [their] trains to travel the net; (ii) maintains [etc.] stations, lines and junctions – including monitoring the train traffic and controlling rail net signals, etc.

65. Rail net aggregates carry traffic: we may model this by associating with each train its [current] status: whether in operation or not, if in operation then its position on its current journey, etc., etc. [We omit details.]

66. Etcetera, etc. !

type
65. RNATraffic = TI $\overrightarrow{m}$ ... [PROGRAMMABLE]
66. ...
value
65. **attr**_Traffic: TA → PHist

Aggregate of Trains

We introduce a notion of train companies[14].

67. With the aggregate of trains we can, for example, associate a map which to each train associate the one train company that "owns" it.

68. Train companies have unique identification (name, etc.).

69. Etc., etc.

[14] `NSF, Union Pacific, CSX, Norfolk Southern, SNCF, Avanti West Coast, DB, SJ, NSB, DSB, Trenitalia, Renfe, Flixtrain, ÖBB, SBB, SNB, NS, ...`

type
67. ATTICI = TI $\overrightarrow{m}$ CI [PROGRAMMABLE]
68. CI = ...
value
67. **attr**_TICI: AT → TICI

Intentional Pull

2.3 Perdurants

In this section we "lift", by transcendental deduction, manifestable parts "into" behaviours.

2.3.1 Communication

70. Behaviours synchronize and communicate messages, `MSG`

71. over channels.

type
70. MSG
channel
71. { **comm**[{i,j}] | i,j:UID • {i,j} ⊆ σ_{uid} } MSG

2.3.2 Actions

Narration & formalization of actions are integral parts of behaviour definitions. Here we shall just briefly "name" train, line, junction, station and passenger actions.

72. **Train Actions:** Trains record (i) moving along the rail net: entering and leaving stations, lines and junctions, as well as (ii) taking passengers aboard and their disembarking at stations.

73. **Line Actions:** Lines record trains entering, moving along and leaving lines.

74. **Junction Actions:** Junctions record trains entering, moving along and leaving junctions.

75. **Station Actions:** Stations record trains entering, moving along and leaving stations. – as well as

76. **Passenger Actions:** Passengers record their getting on and off trains.

[14] `LEvt` to be defined

2.3.3 Behaviours

Signatures

77. The railway system behaviours, beside the identity and mereology have the following attribute signatures:

    (a) **Trains:**

    (b) **Lines:**

    (c) **Junctions:**

    (d) **Stations:**

    (e) **Passengers:**

value
77a. train: TI → MT → (TServ×TPos×TMS×TLoad×THist)
77b. line: LI → ML → (LLen×LΩ) → (LΣ×LHist)
77c. junction: JI → MJ → (JLen×JOmega) → (JΣ×JHist)
77d. station: SI → MS → (SLen×SΩ) → (JΣ×JHist)
77e. passenger: PI → MP → (...) → (PState×PTickets×PHist)

Definitions

78. **Train** behaviours internal non-deterministically, ⌈⌉, alternates between

    (a) being *at* a station, holding at a *platform*, unloading or taking passengers aboard;

    (b) *passing* through a station without stop;

    (c) *moving* along a *line*, possibly halting for a while;

    (d) *moving* along a *line*, *entering* a junction or a station;

    (e) *moving* along a *junction*, entering a *line* or a *station*.

value
78. train(ti)(sjlis)(,thist) ≡
78a. train_at_platform(ti)(sjlis)(ts,**mk**_inS(si,pid),tms,tload,thist)
78b. ⌈⌉ train_passing_station(ti)(sjlis)(ts,**mk**_inS(si,_),tload,thist)
78c. ⌈⌉ train_moving_along_line(ti)(sjlis)(ts,**mk**_onL(sji,f,jsi),tload,thist)
78d. ⌈⌉ train_at_line_entering_next_unit(ti)(sjlis)(ts,**mk**_onL(sji,1,jsi),tload,thist)
78e. ⌈⌉ train_in_junction_entering_next_unit(ti)(sjlis)(ts,**mk**_inJ(ji),tload,thist)

79. **Line** behaviours external non-deterministically, [], alternates between

    (a) receiving (line event) messages from trains (the time and the train position: (i) entering the line, (ii) moving along, (iii) stopping, for a moment, on the line, and (iv) leaving the line).

    (b) The line behaviour records these messages in its line history.

value
79. line(li)(ml:(sjis,tis))(llen,lω)(lσ,lhist) $\equiv$
79a. **let** (τ,levt) = $\lbrack\rbrack$ { **comm**[{li,ti}] **?** | ti:TI • ti $\in$ tis} **in**
79b. line(li)(ml)(llen,lω)(lσ,$\langle$(τ,levt)$\rangle$$\widehat{}$lhist) **end**

80. **Junction** behaviours external non-deterministically, $\lbrack\rbrack$, alternates between
    (a) receiving (junction event) messages from trains (the time and the train position: passing through the junction.
    (b) The junction behaviour records these messages in its junction history.

value
80. junction(ji)(mj:(sjis,tis))(jlen,jω)(jσ,jhist) $\equiv$
80a. **let** (τ,jevt) = $\lbrack\rbrack$ { **comm**[{ji,ti}] **?** | ti:TI • ti $\in$ tis} **in**
80b. junction(li)(ml)(llen,lω)(lσ,$\langle$(τ,jevt)$\rangle$$\widehat{}$lhist) **end**

81. **Station** behaviours
    (a)
    (b)
    (c)
    (d)
    (e)

value
81. station(si)(ms)(slen,sps,sω)(ssigma,shist) $\equiv$
81a.
81b.
81c.
81d.
81e.

82. **Passenger** behaviours
    (a)
    (b)
    (c)
    (d)
    (e)

value
82. passenger(pi)(mp)(psta,pticks,phist) $\equiv$
82a.
82b.
82c.
82d.
82e.

Initialization

83. (a)
    (b)
    (c)
    (d)
    (e)

value
83.
83a.
83b.
83c.
83d.
83e.

3 Conclusion

References

[1] Dines Bjørner. Formal Software Techniques in Railway Systems. In Eckehard Schnieder, editor, *9th IFAC Symposium on Control in Transportation Systems*, pages 1–12, Technical University, Braunschweig, Germany, 13–15 June 2000. VDI/VDE-Gesellschaft Mess– und Automatisieringstechnik, VDI-Gesellschaft für Fahrzeug– und Verkehrstechnik. Invited talk.

[2] Dines Bjørner. Dynamics of Railway Nets: On an Interface between Automatic Control and Software Engineering. In *CTS2003: 10th IFAC Symposium on Control in Transportation Systems*, Oxford, UK, August 4-6 2003. Elsevier Science Ltd. Symposium held at Tokyo, Japan. Editors: S. Tsugawa and M. Aoki. `www2.imm.dtu.dk/~dibj/ifac-dynamics.pdf`.

[3] Dines Bjørner. New Results and Trends in Formal Techniques for the Development of Software for Transportation Systems. In *FORMS2003: Symposium on Formal Methods for Railway Operation and Control Systems*. Institut für Verkehrssicherheit und Automatisierungstechnik, Techn.Univ. of Braunschweig, Germany, 15–16 May 2003. Conf. held at Techn.Univ. of Budapest, Hungary. Editors: G. Tarnai and E. Schnieder, Germany. `www2.imm.dtu.dk/~dibj/dines-amore.pdf`.

[4] Dines Bjørner. Railways Systems: Towards a Domain Theory. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2003.

[5] Dines Bjørner. The Grand Challenge – FAQs of the R&D of a Railway Domain Theory. In *IFIP World Computer Congress, Topical Days: TRain: The Railway Domain*, IFIP, Amsterdam, The Netherlands, 2004. Kluwer Academic Press.

[6] Dines Bjørner. The TRain Topical Day. In *Building the Information Society, IFIP 18th World Computer Congress, Topical Sessions, 22–27 August, 2004, Toulouse, France — Ed. Renéne Jacquart*, pages 607–611. Kluwer Academic Publishers, August 2004. A Foreword.

[7] Dines Bjørner. Towards a Formal Model of CyberRail. In *Building the Information Society, IFIP 18th World Computer Congress, Tpical Sessions, 22–27 August, 2004, Toulouse, France — Ed. Renéne Jacquart*, pages 657–664. Kluwer Academic Publishers, August 2004.

[8] Dines Bjørner. Domain Engineering. In Paul Boca and Jonathan Bowen, editors, *Formal Methods: State of the Art and New Directions*, Eds. Paul Boca and Jonathan Bowen, pages 1–42, London, UK, 2010. Springer.

[9] Dines Bjørner. Domain Science & Engineering – *From Computer Science to The Sciences of Informatics, Part I of II: The Engineering Part*. *Kibernetika i sistemny analiz*, 2(4):100–116, May 2010.

[10] Dines Bjørner. Domain Science & Engineering – *From Computer Science to The Sciences of Informatics Part II of II: The Science Part*. *Kibernetika i sistemny analiz*, 2(3):100–120, June 2011.

[11] Dines Bjørner. Manifest Domains: Analysis & Description `www.imm.dtu.dk/~dibj/2015/faoc/faoc-bjorner.pdf`. *Formal Aspects of Computing*, 29(2):175–225, March 2017. Online: 26 July 2016.

[12] Dines Bjørner. Domain analysis & description - the implicit and explicit semantics problem `www.imm.dtu.dk/~dibj/2017/bjorner-impex.pdf`. In Régine Laleau, Dominique Méry, Shin Nakajima, and Elena Troubitsyna, editors, Proceedings Joint Workshop on *Handling IMPlicit and EXplicit knowledge in formal system development (IMPEX)* and *Formal and Model-Driven Techniques for Developing Trustworthy Systems (FM&MDD)*, Xi'An, China, 16th November 2017, volume 271 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–23. Open Publishing Association, 2018.

[13] Dines Bjørner. To Every Manifest Domain a CSP Expression `www.imm.dtu.dk/~dibj/2016/mereo/mereo.pdf`. *Journal of Logical and Algebraic Methods in Programming*, 1(94):91–108, January 2018.

[14] Dines Bjørner. Domain Analysis & Description – Principles, Techniques and Modelling Languages. `www.imm.dtu.dk/~dibj/2018/tosem/Bjorner-TOSEM.pdf`. *ACM Trans. on Software Engineering and Methodology*, 28(2):1–67, April 2019. 68 pages.

[15] Dines Bjørner. *Domain Science & Engineering – A Foundation for Software Development*. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg, Germany, 2021. A revised version of this book is [16].

[16] Dines Bjørner. Domain Modelling – A Primer. A short and significantly revised version of [15]. xii+202 pages[15], May 2023.

[15]This book is currently being translated into Chinese by Dr. Yang ShaoFa, IoS/CAS (Institute of Software, Chinese Academy of Sciences), Beijing and into Russian by Dr. Mikhail Chupilko and his colleagues, ISP/RAS (Institute of Systems Programming, Russian Academy of Sciences), Moscow

[17] Dines Bjørner. Domain Analysis & Description. Technical report, Technical University of Denmark, Fredsvej 11, DK-2840 Holte, September 2025.

[18] Dines Bjørner. Domain analysis & description – a tutorial. In *ICTC 2025 Proveedings*, Lecture Notes in Computer Science. Springer, November 2025.

[19] Dines Bjørner, Chris W. George, and Søren Prehn. Scheduling and Rescheduling of Trains. Research Report 52, UNU/IIST, P.O.Box 3058, Macau, December 1995. .

[20] Dines Bjørner, Chris W. George, and Søren Prehn. Computing Systems for Railways — A Rôle for Domain Engineering. Relations to Requirements Engineering and Software for Control Applications. In *Integrated Design and Process Technology. Editors: Bernd Kraemer and John C. Petterson*, P.O.Box 1299, Grand View, Texas 76050-1299, USA, 24–28 June 2002. Society for Design and Process Science. `www2.imm.dtu.dk/~dibj/pasadena-25.pdf`.

[21] Dines Bjørner, C.W. George, B.Stig Hansen, H. Laustrup, and S. Prehn. A Railway System, Coordination'97, Case Study Workshop Example. Research Report 93, UNU/IIST, P.O.Box 3058, Macau, January 1997. .

[22] Dines Bjørner, Dong Yu Lin, and Søren Prehn. Domain Analyses: A Case Study of Station Management. Research Report 23, UNU/IIST, P.O.Box 3058, Macau, 9 November 1994. Presented at the *1994 Kunming International CASE Symposium*: KICS'94, Yunnan Province, P.R.of China, 16–20 November 1994. .

[23] Dines Bjørner and Martin Pěnička. Towards a **TRAIN** Book for **T**he **RAI**lway Domai**N**. Techn. reports, Technical University of Denmark, Fredsvej 11, DK-2840 Holte, Denmark, August 2004. `http://www.imm.dtu.dk/~dibj/train-book.pdf`.

[24] Dines Bjørner, Søren Prehn, and Chris W. George. Formal Models of Railway Systems: Domains. Technical report, Dept. of IT, Technical University of Denmark, Bldg. 344, DK–2800 Lyngby, Denmark, September 23 1999. Presented at the FME Rail Workshop on Formal Methods in Railway Systems, FM'99 World Congress on Formal Methods, Toulouse, France. Avaliable on CD ROM.

[25] Martin Pěnička and Dines Bjørner. From Railway Resource Planning to Train Operation — a Brief Survey of Complementary Formalisations. In *Building the Information Society, IFIP 18th World Computer Congress, Topical Sessions, 22–27 August, 2004, Toulouse, France — Ed. Renéne Jacquart*, pages 629–636. Kluwer Academic Publishers, August 2004.

[26] Martin Pěnička, Albena Kirilova Strupchanska, and Dines Bjørner. Train Maintenance Routing. In *FORMS'2003: Symposium on Formal Methods for Railway Operation and Control Systems*. L'Harmattan Hongrie, 15–16 May 2003. Conf. held at Techn.Univ. of Budapest, Hungary. Editors: G. Tarnai and E. Schnieder, Germany. `www2.imm.dtu.dk/~dibj/martin.pdf`.

[27] Albena Kirilova Strupchanska, Martin Pěnička, and Dines Bjørner. Railway Staff Rostering. In *FORMS2003: Symposium on Formal Methods for Railway Operation and Control Systems*. L'Harmattan Hongrie, 15–16 May 2003. Conf. held at Techn.Univ. of Budapest, Hungary. Editors: G. Tarnai and E. Schnieder, Germany. `www2.imm.dtu.dk/~dibj/albena.pdf`.

[28] Yu Xinyiao. Stability of Railway Systems. Technical Report 28, UNU/IIST, P.O.Box 3058, Macau, May 1994.

[29] Chao Chen Zhou and Huiqun Yu. A Duration Model for Railway scheduling. Technical Report 24b, UNU/IIST, P.O.Box 3058, Macau, May 1994.

A Index

There are 169 definitions.