Subject: How to model 3 topology levels in rail.ML
Posted by Thomas Langkamm on Fri, 26 Apr 2019 10:19:59 GMT
View Forum Message <> Reply to Message

Hello all,

I'm working at PSI Transcom. We learned about the NEAT infrastructure editor that will export rail.ML, and are very interested in using it as an external editor for our infrastructure using rail.ML 3.x. I'm checking out how to model all objects we need in rail.ML, and came across a little problem that I'd like to share: How could we model 3 or more topology levels?

We create track plans with all details (microscopic rail.ML model), but we usually import our timetables from planning programs and need to match the data from the planning programs to our track plans. Many planning programs use a 2-tier topology: One that is based on stations, and one that is more detailed (but not as detailed as the microscopic topology), usually on basis of station tracks and possibly parking tracks. This 2-tier topology is necessary, as timetable programs need to track connectivity and changes of direction in more detail than a station-based topology allows: Changes of direction can't be checked at all in a station based topology, and connectivity within a station is a thing because it might take passengers (or drivers switching to a different train journey) between a few seconds to get to a train on the other side of the same platform or several minutes to get to a different platform.

For example, a track plan like this

which has a basic A-B-C mesoscopic (station) topology could look like this in the "station track" based topology:

or

Connectivity is not induced by the technical ability to move between 2 station, but by what's in the planned timetable. For example, trains may be able to go directly from A2 to P3, but the edge may be missing because this is not something that will be planned in advance.

Now, if we get a timetable with a train journey ending at B, we get stations (and times) like C1 -> B2 -> P3, and need to match these locations to our track plan.

As far as I understand, rail.ML 3.1 currently defines 2 topology levels: A mesoscopic topology (station based) and a microscopic topology (full details). But how would we include an intermediate topology (that must be consistent with the other 2)?

Structurally I have some intuition how this could work, and work in a generic way: Each element is part of a (named) topology, and there can be inclusions. We can name the 3 or more standard topologies that we want to standardize (mesoscopic = nodes are stations, microscopic = nodes are netElements/track segments, and [insert cool name here] = nodes correspond to timetable topology), assign nodes (netElements) to topologies, and allow for inclusions. Inclusions can be

0..1 : 0..n (or 0..1 : 1..n in a complete model, where any element of a high-level topology has at least one element in a more detailed topology).

Thoughts?

Best regards, Thomas

Subject: Re: How to model 3 topology levels in rail.ML
Posted by christian.rahmig on Fri, 03 May 2019 10:00:12 GMT

View Forum Message <> Reply to Message

Dear Thomas,

welcome to the railML forum!

Am 26.04.2019 um 12:19 schrieb Thomas Langkamm:
> [...] How could we model 3
> or more topology levels?

railML 3.x topology model is based on RailTopoModel and very generic in
application. This means that you can model as many topology levels as
you want. However, in discussion we came to the conclusion that three
levels are enough: micro, meso and macro.

>
> We create track plans with all details (microscopic rail.ML
> model), but we usually import our timetables from planning
> programs and need to match the data from the planning
> programs to our track plans. Many planning programs use a
> 2-tier topology: One that is based on stations, and one that
> is more detailed (but not as detailed as the microscopic
> topology), usually on basis of station tracks and possibly
> parking tracks. [...]

Micro topology level is used to model the track network in detail. The
timetable view on stations and lines can be modelled as macroscopic
topology. In between, you can define the mesoscopic topology, in which
you aggregate the line tracks to lines, but remain the station track
network. This way of aggregating is different to the approach used in
the Simple Example where the station tracks have been aggregated and the
line track remains. However, both approaches are possible from schema
point of view, because so far there is no exact definition of the terms
"micro", "meso" and "macro".

Nevertheless, there is an important rule to follow:
The aggregation of a more detailed topology level towards the more

aggregated topology level must be unambiguous in order to ensure
bi-directional level conversion. This is ensured with the
<elementCollection*> elements, which list the (detailed) elements
belonging to the (aggregated) element.

Does this answer help you for the moment?

@community: what do you think about the definition of the topology
levels "micro", "meso" and "macro"?

Best regards
Christian

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany   www.railml.org

Subject: Re: How to model 3 topology levels in rail.ML
Posted by Thomas Langkamm on Fri, 07 Jun 2019 08:44:27 GMT
View Forum Message <> Reply to Message

I guess part of my confusion came from the rail.ML simple tutorial, where there is no clear distinction between mesoscopic and macroscopic topology and two tracks belonging to different platform edges are grouped to one mesoscopic netElement. In my world, the mesoscopic topology is precisely the one where "timetable-relevant" netElements correspond 1:1 to the platform edges.

I guess if I switch meso to macro in the tutorial, then I understand how to model topologies in rail.ML :)

Maybe I should define more precisely what I mean with mesoscopic topology, to check that we are talking about the same thing. When grouping microscopic netElements to mesoscopic netElements, I would follow these rules:

In open areas (outside of stations/operational points), the modelliung is up to the user. If we consider simple tracks between stations, we may have only one netElement for a single track and two netElements if we have double tracks. More elements if we have a more complex connectivity.
Within stations we have to distinguish between tracks that have an associated platform edge, and the remaining tracks. All netElements incident to one platform edge will be grouped to one mesoscopic netElement, so we end up with exactly one mesoscopic netElement for every per track visible in a printed timetable.
For parking tracks we'll have either one mesoscopic netElement for a group of parking tracks, or one mesoscopic netElement per track in the depot.

Maintenance areas are handled similar to parking areas.
For the remaining tracks, any modelling is feasible as long as it maintains the correct connectivity between the previously defined mesoscopic netElements.

As for 3 being enough, on that level of detail I think we're right. But there may be other topologies on top of macro, for example grouping operational points to areas (perhaps controlled by interlockings, perhaps belonging to different regions or countries).
