Subject: railML 3.2: Additional information for travel paths in a macroscopic netElement
Posted by Thomas Langkamm on Mon, 06 Dec 2021 09:27:36 GMT
View Forum Message <> Reply to Message

In railML 2.5 we introduced a new "relation" element to ocps (Adding turning resistances to the <ocp>, https://trac.railml.org/ticket/413). We had some discussions in the IS SCTP group on how to integrate this in railML 3 (which is structurally a bit different from 2.5 there), or if we need this at all.

Here is my proposed solution.

The typical scenario would be that some software works on a mesoscopic or macroscopic version of the network, without detailed knowledge of the microscopic network. We often need more information than the connectivity (which is covered by netRelations) namely information about changes of direction (tracking vehicle position&direction in a train formation) and possibly more information that can be used for routing.

Before I propose a model, I would like to give some background information.

Change of direction (COD)
==========
CODs are considered to be fundamental information in many contexts. This information comes in 2 flavors: (1) Orientation changes (do we reverse driving direction, and are front/back vehicles exchanged?), and (2) the exact number of CODs. An orientation change occurs if we have an odd number of CODs, while the orientation remains the same if that number is even. A few examples where we need one of this information:

** Passenger information systems typically need to know about orientation changes, because the order of the vehicles is reversed. ("1st class in sections A and B today.")
** A duty scheduling system, however, will need to know the precise number of CODs, as each cabin change takes time or needs an additional driver -- we may have 2 alternatives for a train movement, for example one that involves 1 COD and another involving 3 CODs.
** Vehicle dispatch or rostering often only need the same information -- for example the locomotive must be in front of the passenger railcars. However, there are situations where the number of CODs is important as well: Say we have railcars with driver cabins on both side and the cabin on one side has a minor technical issue such that it can't be used to in front, but it can be used as last railcar in a train for example. In this case we need to know if any CODs occur even if the orientation remains (even number of CODs).

These software systems will often operate on a mesoscopic/macroscopic model, i.e. without a full network model, so they can't calculate the CODs on their own. The travel path in a net element may be a black box and hidden from these software systems. We need to provide more information in the infrastructure.

How could we integrate CODs in a macroscopic railML 3.2 model?

Even though we already have a "relation" between a pair of net elements in railML 3 (netRelation), this element is not sufficient to store this information because we need the context of a 3rd net element. Consider the following track plan for example. C is a station with a one siding track where trains could be parked.

With a microscopic netElement structure as follows:

This could have a macroscopic netElement structure as follows:

(Obviously we could split C in 2 net elements, which might be a better design. However, let's stick with one net element because the type of problems that we will see might arise in more complex models where we don't have the luxury to split net elements.)
Note that all netRelations in the macroscopic model are navigable in both directions, and we can navigate between C and all other net elements without changing orientation. This of course changes if we consider triples of net elements:

* From A to B via C (A-C-B) we always have a COD. The route may or may not involve shunting, depending on whether we use the switch on the left or the siding track to the right of C.
* For A-C-E we have a direct path (no change of direction or shunting. But we could also go via the top platform and then use2 CODs to get to E.
* A-C-D is similar to A-C-E (even though the platforms are reversed for the direct and indirect connection).
* For B-C-D we have 2 CODs and possibly shunting. Thus, the orientation of the train remains identical but a driver would have to do 2 cabin changes.
* B-C-E has only a direct connection. (We could go via the top platform in C, but then we have a circle in the path as we pass the bottom platform twice. IMO an internal path must not contain a circle.)
* D-C-E has a change of orientation and either 1 COD or 3 CODs.

Assuming that the model allows no CODs within a net element (which IMO is a reasonable assumption if we want to track CODs at all), we have a well defined information if a train changes direction or not when moving from X to Z via Y that does not depend on the travel route within these net elements. The _number_ of CODs (number of cabin changes) may depend on the travel route within the net elements -- but we do know if that number is even or odd regardless of the travel route.

Therefore I would suggest to use an extension of what we have in railML 2.5 where we use "changeDrivingDirection". I believe we should be able to add more information, namely if the

orientation is changed (changeDrivingDirection=true could also imply 2 CODs resulting in no orientation change) as well as the precise number of CODs (optional). I also believe that the "type" attribute should be an enumeration, that is, we have true/false for each of the proposed types. (requiresShunting, changeDrivingDirection and crossesContraflowTraffic could each independently be true or false.)

We add an object "travelPathInformation" (or maybe a better name? I don't want to use "route" as this would be ambiguous) that refers to 3 netElements: A start X, an intermediary Y and an endpoint Z. (Alternatively we could reference the netRelations linking X/Y and Y/Z in this order.) It describes the internal path for a train coming from X and moving to Z, within the net element Y. There can be more than one travelPathInformation for a sequence X-Y-Z if there is more than one internal path a train could take.

The object has the following attributes (besides the usual attributes like ID, designators and such):
* directed: boolean (mandatory). If true, there might be another travelInformation element for the route C-B-A. If false, the information here can be used for A-B-C as well as for C-B-A.
* distance: numeric. Travel distance for a train using this route (meters travelled on track which is well defined).
* lengthRestriction: numeric. Maximum length of the train for this route (for any change of direction we may have a length limit).
* changesOrientation: boolean. If true, we have a net change of orientation (last railcar coming from A moving to B is now the first railcar on the way to C).
* numberOfDirectionChanges: int. Describes how often the train changes direction.
* requiresShunting: boolean. (Same as railML 2.5.)
* crossesContraflowTraffic: boolean. (Same as railML 2.5.)

Here is how this could look for the travelPathInformations within C (listing all objects starting with A and ending in B, D or E):

<travelPathInformation netElement="neC" start="neA" destination="neB" directed="false" distance="2000" lengthRestriction="200" changesOrientation="true" numberOfDirectionChanges="1" requiresShunting="false" crossesContraFlowTraffic="false"/>
<travelPathInformation netElement="neC" start="neA" destination="neB" directed="false" distance="2400" lengthRestriction="300" changesOrientation="true" numberOfDirectionChanges="1" requiresShunting="true" crossesContraFlowTraffic="false"/>
<travelPathInformation netElement="neC" start="neA" destination="neE" directed="false" distance="3000" lengthRestriction="200" changesOrientation="false" numberOfDirectionChanges="0" requiresShunting="false" crossesContraFlowTraffic="false"/>
<travelPathInformation netElement="neC" start="neA" destination="neE" directed="false" distance="3400" lengthRestriction="200" changesOrientation="false" numberOfDirectionChanges="2" requiresShunting="true" crossesContraFlowTraffic="false"/>
<travelPathInformation netElement="neC" start="neA" destination="neD" directed="false" distance="2500" lengthRestriction="200" changesOrientation="false" numberOfDirectionChanges="0" requiresShunting="false" crossesContraFlowTraffic="false"/>
<travelPathInformation netElement="neC" start="neA" destination="neD" directed="false" distance="2900" lengthRestriction="200" changesOrientation="false" numberOfDirectionChanges="2" requiresShunting="true" crossesContraFlowTraffic="false"/>

(If we make this a child of netElement neC, then the netElement="neC" reference is not necessary.)

Routing
=======
This was given as the main application for the railML 2.5 addition. Basically, we want to determine the route(s) of a train journey where start and endpoint are known but there are several possible routes. However, routing happens on many levels and might need further information. If we want to plan train journeys (even in an early planning stage), I would expect that we need more detailled information than just a "turning resistance" in the network model. And I'm not sure if the infrastructure model is the right place to handle these informations.

As far as I'm concerned I wouldn't include "averageDelayTime" (from railML 2.5), because the time may be highly dependent on other factors (like length and type of the train, if we have a change of direction and the driver has to change cabins). Also "delay" assumes that there is some basis to calculate the delay on, and it's not clear how this basis is defined.

If we want to allow some form of routing preference, I suggest to use a numeric attribute "weight", or possibly an array of weights (weight/id combination), allowing to give a preference that would be optimizer friendly. (In railML we have "priority", but that's an integer between 1 and 255 which might be a problem for optimizers.) This allows to pass some data to an optimizer and use the available paths within that net element in a routing algorithm. The precise meaning of the weight would have to be described on a per-case bases.

This model allows us to migrate a railML 2.5 schema to 3.2, by using 2 weights instead of "averageDelayTime" and "priority". This is how we could implement this:

<travelPathInformation netElement="neC" start="neA" destination="neB" directed="false" distance="2000" lengthRestriction="200" changesOrientation="true" numberOfDirectionChanges="1" requiresShunting="false" crossesContraFlowTraffic="false">
<weight id=" delayTime" value="1.5"/>
<weight id="priority" value="1"/>
</travelPathInformation>
<travelPathInformation netElement="neC" start="neA" destination="neB" directed="false" distance="2400" lengthRestriction="300" changesOrientation="true" numberOfDirectionChanges="1" requiresShunting="true" crossesContraFlowTraffic="false">
<weight id="delayTime" value="6.0"/>
<weight id=" priority" value="20"/>
</travelPathInformation>

Subject: Re: railML 3.2: Additional information for travel paths in a macroscopic netElement
Posted by christian.rahmig on Sun, 20 Feb 2022 07:22:34 GMT

View Forum Message <> Reply to Message

Dear Thomas,

thank you very much for your extensive proposal for the travel paths in macroscopic nodes. The issue is on the table for implementation with upcoming railML 3.2 [1].

@Dear Community, what do you think about this proposal? What are your requirements to be considered in the final solution?

[1] https://development.railml.org/railml/version3/-/issues/452

Best regards
Christian

Subject: Re: railML 3.2: Additional information for travel paths in a macroscopic netElement
Posted by on Tue, 22 Feb 2022 10:01:47 GMT
View Forum Message <> Reply to Message

Dear Thomas and Christian,

I think this is very much related to the topic "Reversing trains and formations" which is already dealt with at <timetable> in railML 2.x [1]. The need for a Change of direction information is imho commonly agreed.

Currently, I do not see why there should be a greater problem to deal with this under railML 3.x than at 2.x. So, I would expect we get a similar solution at 3.x as we use in 2.x, a solution which gives the COD information at a certain train or train path under <timetable>.

If you see a crucial reason why this should be handled at <IS> rather than at <TT>, or may be I misunderstood the main point, please enlight me.

Best regards,
Dirk.

[1] https://wiki2.railml.org/wiki/Dev:Reversing_trains_and_forma tions

---

Subject: Re: railML 3.2: Additional information for travel paths in a macroscopic netElement
Posted by christian.rahmig on Mon, 07 Mar 2022 11:09:09 GMT
View Forum Message <> Reply to Message

Dear all,

apart from the solution proposed by Thomas and the timetable related approach mentioned by

Dirk, I want to suggest a very basic change in the RTM foundation. Currently, a <netRelation> connects two <netElement> objects via child elements <elementA> and <elementB>. Based on discussions related to the driving directions in macroscopic nodes, I propose to adapt the model for enabling connections of more than two <netElement> objects. In particular, the approach foresees:
* renaming elements <elementA> and <elementB> into <fromElement> and <toElement>
* adding new optional, repeatable child element <viaElement> (0..*)

In order to keep topology simple, there should be no further attributes or features added in a <netRelation>. Consequently, the attributes suggested by Thomas in his approach need to be moved somewhere else. Maybe, a combination with the timetable related ideas from Dirk is feasible.

Dear community, what do you think about this proposal? Any feedback is highly appreciated...

Best regards
Christian

Subject: Re: railML 3.2: Additional information for travel paths in a macroscopic netElement
Posted by Thomas Langkamm on Wed, 09 Mar 2022 15:24:31 GMT
View Forum Message <> Reply to Message

Dear Dirk,

I'm not quite sure that I understand what you're proposing. Could you be more specific? There are several use cases where it's necessary to track the orientation/position of a railcar [possibly in a formation] independent of a timetable, and therefore we can't really use changes of direction supplied in TT.

For example, in conflict management we may have to reroute a train over a different part of the network (a detour) because the original travel path (which could be fairly long, more than just a couple of track routes) is not available. Here it's necessary to know if the train changes orientation because it uses this detour.
In maintainance planning (more specifically, planning of the necessary shunting), a typical case is that a railcar is located at some station and needs to be moved to a workshop at a different station. Assuming that we achieve this by adding the railcar to the end (or front) of existing trains, we need to find a sequence of train journeys that get the train to the workshop but also in a way that shunting is minimized (for example the railcar ends up as first railcar on the overnight parking track). Here it's not practical to look at all train journeys in a timetable, but rather we'll look for a travel path (macroscopic or mesoscopic) first and then find train journeys covering the distance second, hence we need changes of direction in the macroscopic model.
As for the other attributes (that are mostly related to routing) I don't see how they would work in a timetable context either. If we have two possible travel paths between A and B via C (say using different platforms at C) and we want to encode a priority for routing (say one travel path should

only be used in emergencies), there is no timetable context.

Subject: Re: railML 3.2: Additional information for travel paths in a macroscopic netElement
Posted by on Wed, 09 Mar 2022 17:13:36 GMT
View Forum Message <> Reply to Message

Dear Thomas,

as I wrote: "may be I misunderstood the main point".

And, I do not need to be convinced about the necessity of a Change-of-direction-information in general.

It seems that I misunderstood your problem because
- I assumed that infrastructure can always be microscopic if necessary and that
- for some reason you do not take a microscope into account as a solution.

(There is of course no strict border between macro- and microscopic models. In this sense here, a macroscopic model is one with rather no points/switches as linking elements (nodes). A microscopic model is one with rather all points/switches as linking elements (nodes).)

So, can we agree that in general, there is already a solution for your problem: Use a microscopic model of infrastructure!

Don't worry, Thomas, I am also a friend of macroscopic models especially in conjunction with timetables. So, in case I am right with that assumptions, we can try to find a way to encode direction information in macroscopic infrastructure models.

In my view, this would have to be matrix for each (macroscopic) junction of the network.
(A "macroscopic junction" is a node with more than two edges, so either a station or a junction with at least one branch line.)

Such a matrix for each macroscopic junction would have all incoming and outgoing tracks or lines of the junction as rows and columns.
In each cell of the matrix a Boolean information would be written: Change of Direction = yes or no.
So, a software can read from the matrix for each combination of from-where (incoming) to-where (outgoing) a Change of Direction would be necessary.

Example for station Lillestrøm:

GO: Gardermobanen from/to Oslo
GL: Gardermobanen from/to Lufthavn
HS: Hovedbanen from/to Strømmen
HL: Hovedbanen from/to Leirsund

KF: Kongsvingerbanen from/to Fetsund

	GO	GL	HS	HL	KF
GO	x	-	x	-	-
GL	-	x	-	x	x
HS	x	-	x	-	-
HL	-	x	-	x	x
KF	-	x	-	x	x

x = Change of Direction = yes
- = no Change of Direction

The main diagonal is normally set to "Change of Direction" except for stations with turning loops. The halves above and below the main diagonal are normally symmetrical except for some strange stations with one-way triangles or such.

So, Thomas, if such matrices in <infrastructure> would be a solution for your problem, I would like to support this solution.
I would recommend to make one of such matrix as an optional sub-element of an <ocp> or similar object in railML3. The rows and columns should be line-IDs or track-IDs.

But first, in general, we need to make the decision whether to support a redundancy where a microscopic model is already a solution at all.

Hope I didn't miss the point,
with best regards,
Dirk from Dresden.

Subject: Re: railML 3.2: Additional information for travel paths in a macroscopic netElement
Posted by Thomas Langkamm on Thu, 10 Mar 2022 11:43:25 GMT
View Forum Message <> Reply to Message

Dear Dirk,

So, can we agree that in general, there is already a solution for your problem: Use a microscopic model of infrastructure!

True. But many systems doing passenger information, vehicle dispatch or maintainance planning do not support a microscopic model, only a mesoscopic model (they know the different tracks in a station where a train might stop in a train journey or for parking, but have no detailled information about switches and the rest of the network). In some cases they will work only on a macroscopic model (stations as atoms).

In my view, this would have to be matrix for each (macroscopic) junction of the network.

(A "macroscopic junction" is a node with more than two edges, so either a station or a junction with at least one branch line.)
I don't think that a matrix involving the number of CODs would work. (You need to distinguish between 0 and 2, because not all trains can travel in both directions.) There may be several ways to construct a travel path from A to B (say both being a tuple of track and direction), for example one that involves only one change of direction and another one involving 3 changes of direction. However, the latter one may still be preferrable because it uses a part of the network that is rarely used and therefore the required shunting wouldn't block part of the tracks that need a high availability.

To encode both variants and the other attributes we end up using a solution that is structurally very similar to the one I proposed. Instead of primitives (bool, int or double) in the matrix you would end up with a list of complex entries.

Finally, the matrix doesn't work for pure macroscopic models. Look at my example again: You can go from A to C and from C to E without a change of direction, yet if you go from A to C to E it depends on which platform you use in C if you need 0 or 2 change of directions. The matrix would work only on the mesoscopic levels if we have all platforms. And again, some systems may not have that detail, especially parking areas are often treated as "black boxes" by long-term planning systes.

I'm not at all emotionally tied to my proposal in any way, if there is a better way to do things then I'm all for it. But the solution has to work for systems that have incomplete information about the model. The main purpose IMO is to allow adding information on a mesoscopic/macroscopic leve that can (of course) be derived from the microscopic model, but isn't available because the systems do not support storing/evaluating a microscopic or mesoscopic model.

Subject: Re: railML 3.2: Additional information for travel paths in a macroscopic netElement
Posted by Thomas Langkamm on Thu, 10 Mar 2022 11:53:00 GMT
View Forum Message <> Reply to Message

Dear Christian,

christian.rahmig wrote on Mon, 07 March 2022 12:09
I want to suggest a very basic change in the RTM foundation. Currently, a <netRelation> connects two <netElement> objects via child elements <elementA> and <elementB>. Based on discussions related to the driving directions in macroscopic nodes, I propose to adapt the model for enabling connections of more than two <netElement> objects. In particular, the approach foresees:
* renaming elements <elementA> and <elementB> into <fromElement> and <toElement>
* adding new optional, repeatable child element <viaElement> (0..*)
I think this would be a valid approach if we were working on a new major version of railML and had decided to tweak the netElement/netRelation structure anyway, but in railML 3.x I would strongly prefer storing the information either as new object or somewhere else, but not as part of the netElement/netRelation structure. The reasons for this are as follows:

The netElement and netRelation objects are the very fundament of the model. Even renaming objects should only be considered for new major versions, and should be done only if there are major reasons for it (or against keeping the old structure).

This suggestion does not only rename, but it changes semantics. Right now we can assume that netRelations connect netElements that are adjacent, so the netRelation itself has no extension, travel time or other dimension. The netElement are the places that have an extension where a train can be. Your suggestion changes this, now there could be an extension between fromElement and toElement. This would break any code that, for example, assumes that there is zero travel time and distance between netElements when traversing a netRelation.

Subject: Re: railML 3.2: Additional information for travel paths in a macroscopic netElement
Posted by christian.rahmig on Mon, 14 Mar 2022 14:36:58 GMT
View Forum Message <> Reply to Message

Dear Thomas,

thank you for your feedback. I understand your concerns and I agree with your conclusion to better leave the RTM foundation of topology unchanged. Consequently, only your proposal remains on the list of options.

If there are no objections until 16.03.2022, I am going to implement your solution as summarized in Git issue #452 [1] with one change proposal: rename <travelPathInformation> into <travelPath>. Any last comments from the community are highly appreciated...

[1] https://development.railml.org/railml/version3/-/issues/452

Best regards
Christian