Train Tracking Algorithm

The layout is divided into blocks. A block can contain a single train. A train can span multiple blocks.

The layout hardware may support several methods for detecting trains:

* Block occupancy – The hardware sense that the block is occupied by a train.
* Block edge sensors – The hardware sensors are located on one or more block edges. Two type of block edge sensors are supported
  + Track contact – momentary sensor that senses the passing of a single (or more specific) train points (usually a single point which is near the locomotive’s front).
  + Proximity sensor – senses if a train is above it.

The train tracking algorithm is different for occupancy blocks and block edge sensors. This document describes block edge sensing.

A block can be “virtual block” which means that there is no hardware for detecting if a train is inside it. This is useful to ensure that the block exists for locking purposes, but there is no need to “waste” train detection hardware if the train can never stop in this block.

Currently only track contact-based tracking is implemented. This document describes an algorithm for proximity block edge sensors. As it turns out, this algorithm can replace the current algorithm since track contacts can be for most parts viewed as proximity sensors that sense that proximity of a very short train.

A proximity sensor has two states:

* **Active** – a train is detected above the sensor
* **Not active** – nothing is detected

So basically, we need to describe what should done when an edge sensor become active and what should be done when an edge sensor becomes not active.

An edge block is separation between two blocks. We shall call them *A* and *B*.

Block edge sensor becomes active:

1. If the same train T is found in A and B, then do nothing, nothing really changed, and the edge sensor was not active for some reason.
2. If A contains train T and B is empty – train crossed from A to B (mark B as train’s most recent block).
3. If B contains train T and A is empty – train crossed from B to A (mark A as train’s most recent block)
4. If A and B contains no trains:

For both A and B do the following:

Start with block X (either A or B):

For each block edge of this block (X) that is not a sensor, assume that it was a sensor and it became active, check if a train could have crossed it (using the algorithm that is described here recursively).

1. If still there are no trains that could have crossed, assume that the information about train location is not precise (this is called fuzzy search). This could happen if the user has placed a train on the tracks and did was not precise in indicating where this train is located. In this case check if a train can be found in a block that is close by to A or B (up to distance of 2), if so, relocate the train (assume that this train is located in A if one was found nearby A or B if a train was found nearby B)

Block Edge Sensor becomes non-active

Do the following for each block that contains a train part around the edge that has become non-active:

1. If all the edges of this block are not active, the train part can be removed from this block
2. If the result of step 1 is that there is no block that contains a part of this train, the block that was marked as the train’s most recent block (see above) includes the train part.