1. Railway System Specification

Our railway model follows a circular track with n stations and m trains, with $m <= n$. The trains will move in a circular way, stopping at each station. The trains will be controlled by a signaling system that will allow them to approach the current station, stop, and leave the station. The signaling system will be responsible for managing the trains' movement and ensuring that the trains do not collide by controlling train approaching to the station and leaving the station.

When approaching a station, the signaling system will send the approaching train the order to start slowing down. When a train reaches the station, the signaling system will send it the order to stop. Once the train is stopped, the train will open the doors and wait for the passengers to leave and enter the train. After a certain time, doors will close and the train will communicate with the signaling system that it is ready to leave. The signaling system will send the order to leave the station to the train.

A train approaching a station will have to wait for the train in front of it, if any, to leave the station before it can enter the station. Once the train in the station leaves, the signaling system will send the order to the next train to enter the station.

A train stopped in a station cannot leave it if another train is on its way to the next station or if it is approaching it.

2. Railway System Model

2.1. Variables

The model will have the following global variables:

- `trains_state`: an array with the state of each train.
- `trains_station`: an array of integers with the station of each train.
- `train_to_signal`: an array of channels used to communicate between trains and signals. They are used to inform the signals that the train is approaching the station or ready to leave. The same channel will be used to receive the signal from the signaling system to enter into the station when approching, or to leave the station when stopped. It has two components: the state of the train to indicate if it is approaching or stopped, and the station of the train. The signaling system reuse the same channel to indicate the train that it has green light to enter or leave the station.
- `signal_to_signal`: an array of channels used to communicate between signals to inform whether a train can leave the station or not.

Trains can be in one of the following states: `IN_TRANSIT`, `APPROACHING`, and `STOPPED`. A train also has a variable to store the current station.

2.2. Processes

The railway system model considers two elements, each of which is a Promela process: trains and signals.

2.2.1. Trains

- At the beginning, each train will be in a different station and will be in the stop state. In the `STOPPED` state, and before leaving the station, a train will send a signal `IN_TRANSIT` to the signaling system and wait for the corresponding signal authorising the train to leave the station. That will only happen when there is no train in transit, approaching, or waiting in the next station.

- When the train is transiting to the station, it will move to the approaching state without any condition. Once in the approaching state, the train will signal the signaling system of the current station that it is approaching. The signaling system will allow the train to enter the station and stop if there is no train in the station. The train will remain in the approaching state indicating that is waiting for the station to be clearer until receiving the signal from the signaling system to indicate the train can enter the station.
- When the train is stopped it will signal the signaling system that it is ready to leave the station. The signaling system will check if there is any train waiting to enter the station. If so, it will inform the train to leave the station.

2.2.2. Signals

- Signals will control trains approaching and leaving a station. When the signaling system receives a request from a train to leave the station, it will communicate with the signals in the next station to check if there is a train in transit, approaching, or waiting in the that station. The signal proctype will have two channels to communicate with the next and previous stations, and use local variables to indicate whether there are trains in transit, approaching, or waiting in the station.
- Since signals communicate only with the next and previous stations, they can use control variables to know the state of the trains in the following station when they leave the current station, and clear the variables when they are notified by the signaling system that the train has enter in the following station and therefore the train can leave the current station.
- If there are trains waiting to enter to the next station, the signaling system will not send any signal to the train to leave the station.
- If there is no train in any of those states in the next station, the signaling system will send the signal to the train to leave the station. After a train has left the station the signaling system will inform to any waiting train to enter the station. In that case, it will also check if the previous station had any train waiting to leave. If so, it will inform the train to leave the station.

2.2.3. Init

The `init` process will be in charge of initializing the system by creating the trains and signals proctypes, assigning a unique identifier to each train and signal, and passing to signals the channels that will be used to communicate between them.

It is possible to pass an element of an array to a process as an argument. For example `run signal(signal_to_signal[0])` executes the signal process with the channel `signal_to_signal[0]` as an argument.

2.2.4. Model Structure

The Promela model has the following structure. You can model the trains and signals processes as you consider adequate to fulfill the requirements of the railway system. The provided datatypes are enough to model the system, but if you consider that you need to add more variables, channels, or datatypes, you can. Note that adding more variables will add complexity to the model and the verification process requiring more resources for the verification.

```
#define N 4 //max number of trains
#define M 4 //max number of stations


mtype = {IN_TRANSIT, APPROACHING, STOPPED};

//you must decide the size of the arrays, the type of channels (synchronous/asynchronous),
    and their buffer size for assynchronous channels.

mtype trains_state[] = {STOPPED, STOPPED, STOPPED, STOPPED};

int trains_station[] = {}; //you can assign values statically or dinamically at the
    beginning of the model (in the init process or in the train process)

chan train_to_signal[] = [] of {mtype, int};
chan signal_to_signal[] = [] of {mtype, int};

proctype train(int id){
//define the necessary local variables, if any.
//the train process must be a reactive system that is always travelling from station to
    station.
// you can decide at the beginnig the station where the train is stopped and the state of
    the train.
//after variables have been initialized the train will be in a loop where it will be
    always moving from station to station.

do :: true -> //define the behavior of the train in the different states
od
}

proctype signal(int id, chan signal_next_station, chan signal_previous station){
//define the necessary local variables, if any.
//after variables have been initialized signals will be in a loop waiting for signals from
    trains or other stations.

do :: true -> //define the behavior of the signal in the different states
od

}

init{
  // create the processes with the necessary arguments and right values the following
      lines are just examples.

  run train(0);
  run signal(0, signal_to_signal[0], signal_to_signal[N-1]);
}
```

3. Properties

Assuming we have 4 trains $(0 \cdots 3)$ and 4 stations $(0 \cdots 3)$, specify the following properties using LTL and check them using the Spin model checker.

- If train 0 is waiting to enter in the station 1, it will eventually enter the station.
- It always happens that some time in the future train 1 will transit to station 3.
- It always happens that train 1 will be eventually in a station with an index smaller than the station in which train 4 is stopped.
- it always happens that eventually all the trains will be stopped in some station.
- never can happen that two trains are in the same station in either transit or approaching states.

The syntax of the LTL properties is the following:

```
SYNTAX

Grammar:
  ltl ::= opd | ( ltl ) | ltl binop ltl | unop ltl

Operands (opd):
  true, false, user-defined names starting with a lower-case letter,
  or embedded expressions inside curly braces, e.g.,: { a+b>n }.

Unary Operators (unop):
  []        (the temporal operator always)
  <>        (the temporal operator eventually)
  !         (the boolean operator for negation)

Binary Operators (binop):
  U         (the temporal operator strong until)
  W        (the temporal operator weak until (only when used in inline formula)
  V         (the dual of U): (p V q) means !(!p U !q))
  &&        (the boolean operator for logical and)
  ||        (the boolean operator for logical or)
  /\        (alternative form of &&)
  \/        (alternative form of ||)
  ->        (the boolean operator for logical implication)
  <->        (the boolean operator for logical equivalence)
```