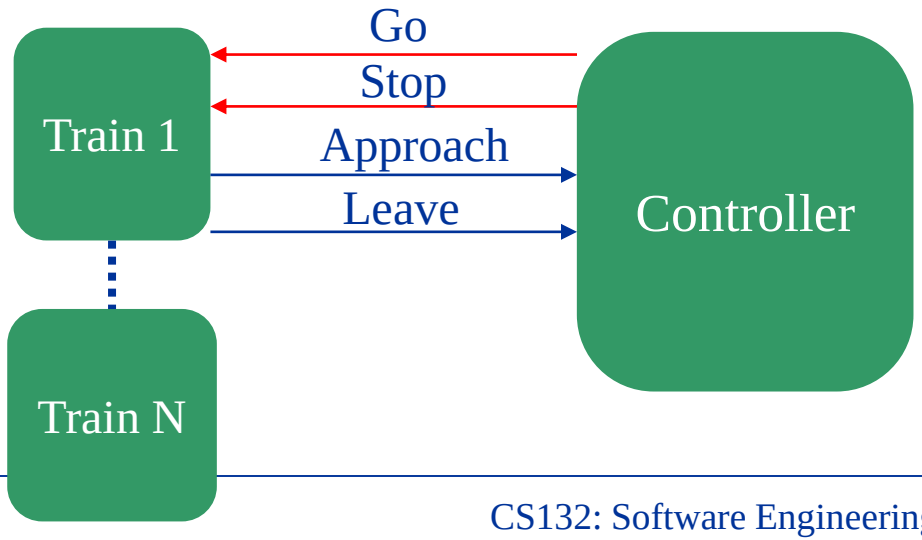
UPPAAL Examples

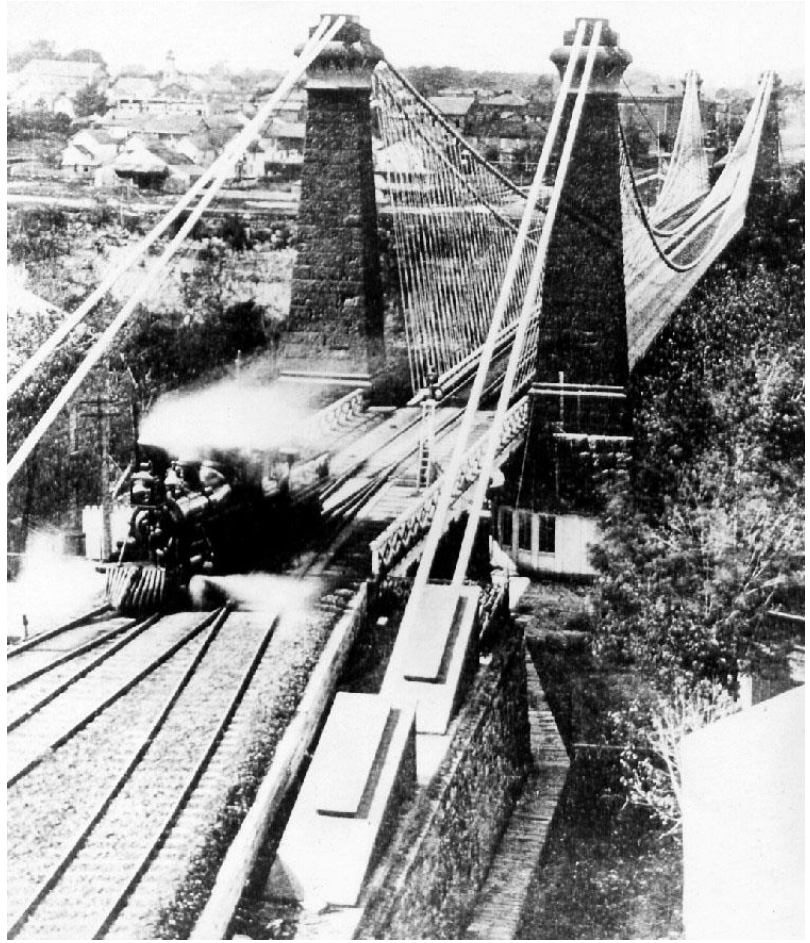
Reference

- Downlaod
  - www.uppaal.org
- Tutorials
  - On the same webpage
  - Recommended:
    - UPPAAL 4.0: Small Tutorial.
    - Uppaal SMC Tutorial

Example: Train-Gate

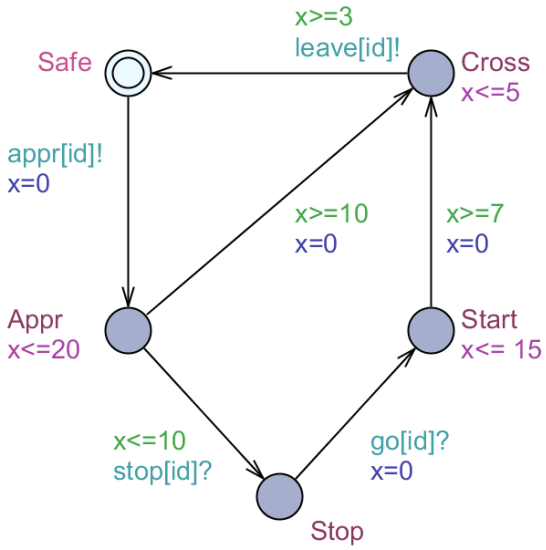
- Niagara Falls Suspension Bridge
- One passage, multiple entries
- Design a software controller that makes sure:
  - Every train arrives the bridge eventually crosses
  - Only one train on the bridge at the same time





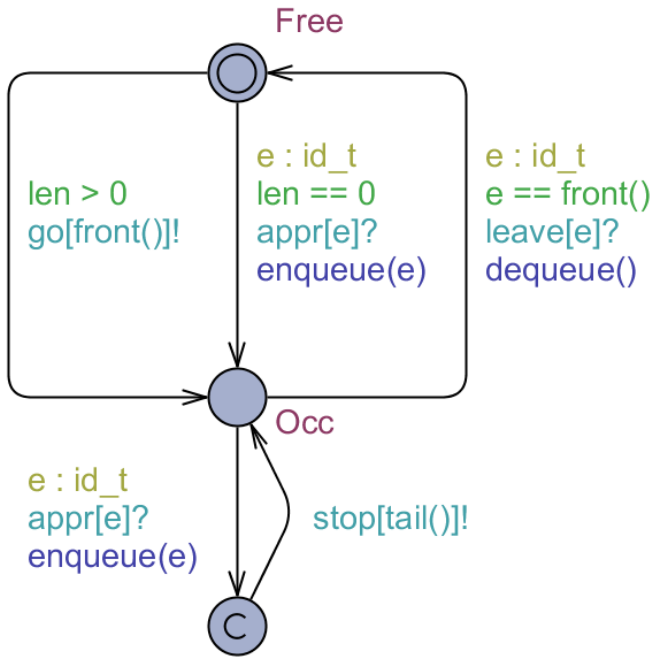
Modeling Trains (Environment)

- Each train has an id
- Each train can approach the gate at any time
- Approaching takes 10-20 sec
- The gate controller can stop a train within 10 sec after its approaching, otherwise the train will cross
- After receive a GO signal, the train will start within 7-15 sec
- Crossing takes 3-5 sec



Example: Train-Gate (cont.)

- Gate controller maintains a queue
- If queue empty and a train approaches, gate stay occupied
- If the gate is occupied and a train approaches, stop the last one in queue
- If the train at the front of the queue leaves, remove it from the queue
- If the gate is free and there are trains in queue, let the front one go




```
typedef int[0,N-1] id_t;

// Put an element at the end of the queue
void enqueue(id_t element)
{
        list[len++] = element;
}
```

```
// Remove the front element of the queue
void dequeue()
{
        int i = 0;
        len -= 1;
        while (i < len)
        {
                list[i] = list[i + 1];
                i++;
        }
        list[i] = 0;
}
```

```
// Returns the front element of the queue
id_t front()
{
   return list[0];
}

// Returns the last element of the queue
id_t tail()
{
   return list[len - 1];
}
```

Example: Train-Gate (cont.)

- Train 0 can eventually cross
  - E<> Train(0).Cross
- Train 0 can be crossing bridge while Train 1 is waiting to cross
  - E<> Train(0).Cross and Train(1).Stop
- Train 0 can cross bridge while the other trains are waiting to cross
  - E<> Train(0).Cross and (forall (i:id-t) i != 0 imply Train(i).Stop)
- There can never be N elements in the queue
  - A[] Gate.list[N] == 0
- There is never more than one train crossing the bridge
  - A[] forall (i:id-t) forall (j:id-t) Train(i).Cross && Train(j).Cross imply i == j
- Whenever a train approaches the bridge, it will eventually cross
  - Train(1).Appr -> Train(1).Cross
- The system is deadlock-free
  - A[] not deadlock