Proposed solution to EX98.3

```
scheme
  RAILWAY =
    class
      type
        TrainId,
        SectionNr = {| n : Nat • n ≤ max |},
        Position′ == single(snr : SectionNr) | double(snr1 : SectionNr, snr2 : SectionNr),
        Position =
          {| p : Position′ • case p of double(s1, s2) → s2 = s1 + 1, _ → true end |},
        Direction == increasing | decreasing

      value
        max : Nat

      type State

      value
        /* observere */
        position : State × TrainId → Position,
        direction : State × TrainId → Direction

      value
        /* afledt observer */
        safe : State → Bool
        safe(σ) ≡ (∀ t1, t2 : TrainId • t1 ≠ t2 ⇒ sections(σ, t1) ∩ sections(σ, t2) = {}),

        sections : State × TrainId → SectionNr-set
        sections(σ, t) ≡
          case position(σ, t) of single(s) → {s}, double(s1, s2) → {s1, s2} end

      value
        /* generatorer */
        move : State × TrainId ~→ State,
        reverse : State × TrainId → State

      axiom
        /* observer−generator aksiomer */
        [ position_move ]
          ∀ σ : State, t, t′ : TrainId •
            position(move(σ, t), t′) ≡
              if t = t′ then
                case direction(σ, t) of
                  increasing →
                    case position(σ, t) of
```

```
                single(s) → if s < max then double(s, s + 1) else single(s) end,
                double(s1, s2) → single(s2)
              end,
            decreasing →
              case position(σ, t) of
                single(s) → if s > 0 then double(s − 1, s) else single(s) end,
                double(s1, s2) → single(s1)
              end
          end
        else
          position(σ, t′)
        end
        pre safe(σ),

  [ direction_move ]
    ∀ σ : State, t, t′ : TrainId • direction(move(σ, t), t′) ≡ direction(σ, t′),

  [ position_reverse ]
    ∀ σ : State, t, t′ : TrainId • position(reverse(σ, t), t′) ≡ position(σ, t′),

  [ direction_reverse ]
    ∀ σ : State, t, t′ : TrainId •
      direction(reverse(σ, t), t′) ≡
        if t = t′ then
          case direction(σ, t) of
            increasing → decreasing, decreasing → increasing
          end
        else
          direction(σ, t′)
        end

value
  safe_move : State × TrainId ~→ State
  safe_move(σ, t) ≡ if safe(move(σ, t)) then move(σ, t) else σ end pre safe(σ)
end
```