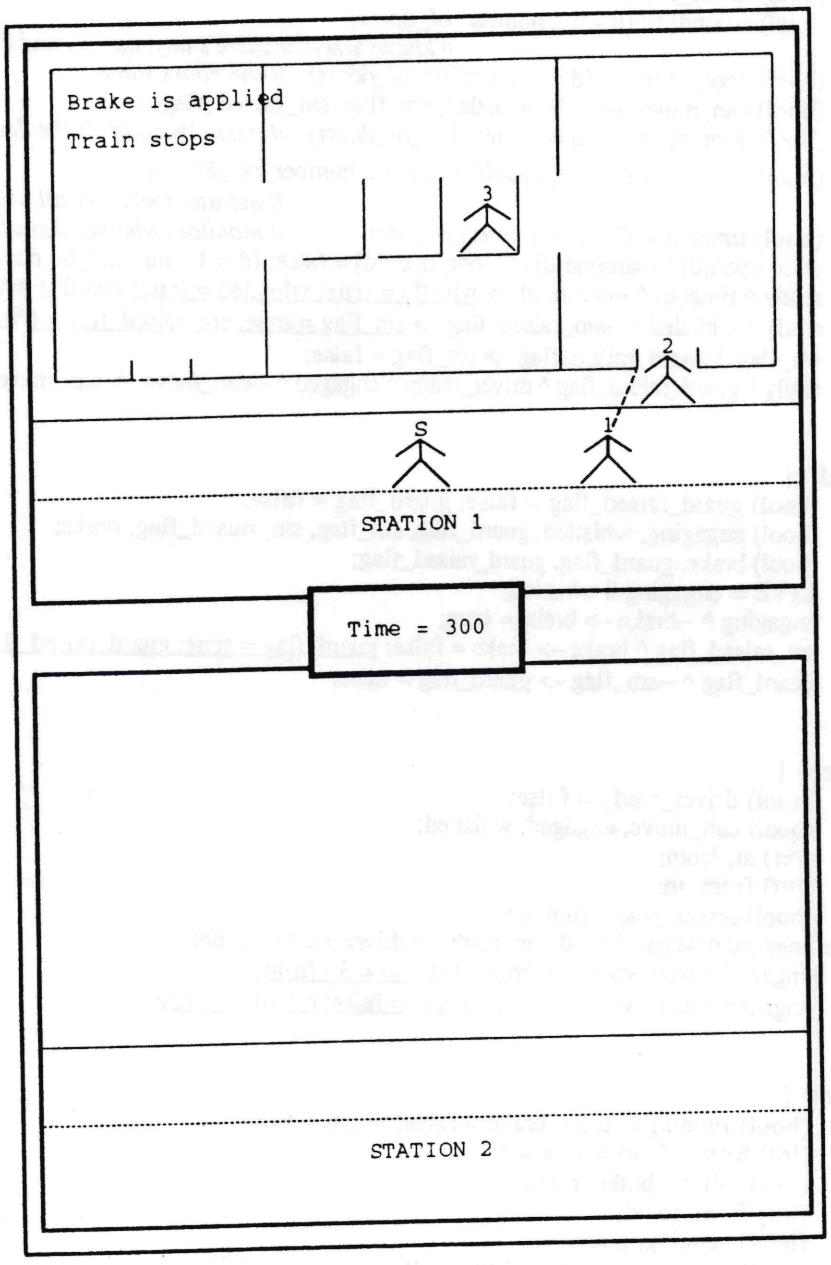
Brake is applied
Train stops
3
2
S
1
STATION 1
Time = 300
STATION 2

```
agent sm() {                                    // The station master:
state       (time) tarrive = ⌈Time⌉;            // registers time of arrival
            (bool) can_move = false;            // determines whether the driver can start the engine
            (bool) whistle = false;             // controls the whistle
            (bool) whistled = false;            // remembers whether he has blown the whistle
            (bool) sm_flag = false;             // controls the flag
            (bool) sm_raised_flag = false;      // remembers whether he has raised the flag
oracle      (time) Limit, Time;                 // knows the time to elapse before departure due
            (bool) guard_raised_flag;           // knows whether the guard has raised his flag
            (bool) driver_ready;                // knows the driver is ready
            (bool) around[d]; (d = 1 .. number_of_doors)
                                                // knows whether there's anybody around doorway
            (bool) door_open[d]; (d = 1 .. number_of_doors)  // the doors status
handle      (bool) can_move, whistle, whistled, sm_flag, sm_raised_flag;
            (bool) door_open[d]; (d = 1 .. number_of_doors)  // partially controls the doors
derivate    (bool) ready = ∧ (¬door_open[d]) | d = 1 .. number_of_doors);
                                                // monitors whether all doors are shut
            (bool) timeout = (Time – tarrive) > Limit;       // monitors whether departure is due
privilege   door_open[d] ^ ¬around[d] -> door_open[d] = false; (d = 1 .. number_of_doors)
            ready ^ timeout ^ ¬whistled -> whistle = true; whistled = true; guard(); whistle = false;
            ready ^ whistled ^ ¬sm_raised_flag -> sm_flag = true; sm_raised_flag = true;
            sm_flag ^ guard_raised_flag -> sm_flag = false;
            ready ^ guard_raised_flag ^ driver_ready ^ engaged ^ ¬can_move -> can_move = true;
}

agent guard() {
state       (bool) guard_raised_flag = false; guard_flag = false;
oracle      (bool) engaging, whistled, guard_flag, sm_flag, sm_raised_flag, brake;
handle      (bool) brake, guard_flag, guard_raised_flag;
derivate    LIVE = engaging || whistled;
privilege   engaging ^ ¬brake -> brake = true;
            sm_raised_flag ^ brake -> brake = false; guard_flag = true; guard_raised_flag = true;
            guard_flag ^ ¬sm_flag -> guard_flag = false;
}

agent driver() {
state       (bool) driver_ready = false;
oracle      (bool) can_move, engaged, whistled;
            (int) at, from;
handle      (int) from, to;
            (bool) driver_ready, running;
privilege   engaged ^ whistled ^ ¬driver_ready -> driver_ready = true;
            engaged ^ from <> at -> from = | at |; to = 3 - from;
            engaged ^ can_move -> driver_ready = false; running = true;
}

agent train() {
state       (bool) running = true; brake = false; alarm = false
            (int) from = 0; to = 1; at = 1;
oracle      (bool) alarm, brake, running;
            (int) from, to, at;
handle      (bool) running, alarm;
derivate    (bool) engaging = running ^ to == at;
            (bool) leaving = running ^ from == at;
            (bool) engaged = ¬running;
privilege   engaging ^ ¬alarm -> alarm = true; guard(); sm();
            leaving ^ alarm -> alarm = false; delete guard(), sm();
            brake ^ running -> running = false;
}
```

Figure 1

```
agent passenger((int) p, (int) d, (int) _from, (int) _to) {
// passenger p is intending to travel from station _from to station _to
// and he will access through door d of the train
state       (int) from[p] = _from;
            (int) to[p] = _to;
            (int) pat[p] = _from;
            (int) door[p] = d;
            (int) pos[p] = 2;
            (bool) alighting[p], boarding[p], join_queue[p,d];
oracle      (int) at, pat[p];
            (bool) queueing[d], pos[p], door_open[d];
handle      (int) pos[p], pat[p];
            (bool) door_open[d];
derivate    alighting[p] = at == pat[p] ^ at == to[p] ^ -2 ≤ pos[p] ≤ 0 ^ engaged;
            boarding[p] = at == pat[p] ^ at == from[p] ^ 0 ≤ pos[p] ≤ 2 ^ engaged;
            join_queue[p,d] = (alighting[p] ^ door_open[d] ^ pos[p] == -1) ||
                              (boarding[p] ^ door_open[d] ^ pos[p] == 1);
            LIVE = ¬(pat[p] == to[p] ^ pos[p] == 2);
privilege   boarding[p] ^ pos[p] == 2 -> pos[p] = 1;
            alighting[p] ^ pos[p] == -2 -> pos[p] = -1;
            alighting[p] ^ ¬door_open[d] -> door_open[d] = true;
            alighting[p] ^ pos[p] == 0 ^ door_open[d] ^ queuing[d]
                              -> pos[p] = 1; pat[p] = latl; pos[p] = 2;
            alighting[p] ^ pos[p] == 0 ^ door_open[d] ^ ¬queuing[d]
                              -> pos[p] = 1; pat[p] = latl; door_open[d] = false; pos[p] = 2;
            boarding[p] ^ ¬door_open[d] -> door_open[d] = true;
            boarding[p] ^ pos[p] == 0 ^ door_open[d] ^ queuing[d]
                              -> pos[p] = -1; pat[p] = at; pos[p] = -2;
            boarding[p] ^ pos[p] == 0 ^ door_open[d] ^ ¬queuing[d]
                              -> pos[p] = -1; pat[p] = at; door_open[d] = false; pos[p] = -2;
}

agent door((int) d) {
state       (bool) queuing[d], occupied[d], around[d];
            (bool) door_open[d] = false;
oracle      (int) pos[p], door[p]; (p = 1 .. number_of_passengers)
            (bool) join_queue[p,d]; (p = 1 .. number_of_passengers)
handle      (int) pos[p]; (p = 1 .. number_of_passengers)
derivate    queuing[d] = there exists p such that join_queue[p,d] == true;
            occupied[d] = there exists p such that (pos[p] == 0 ^ door[p] == d)
            around[d] = there exists p such that (door[p] == d ^ -1 ≤ pos[p] ≤ 1)
privilege   queuing[d] ^ ¬occupied[d] ^ join_queue[p,d] -> pos[p] = 0; (p = 1 .. number_of_passengers)
}
```

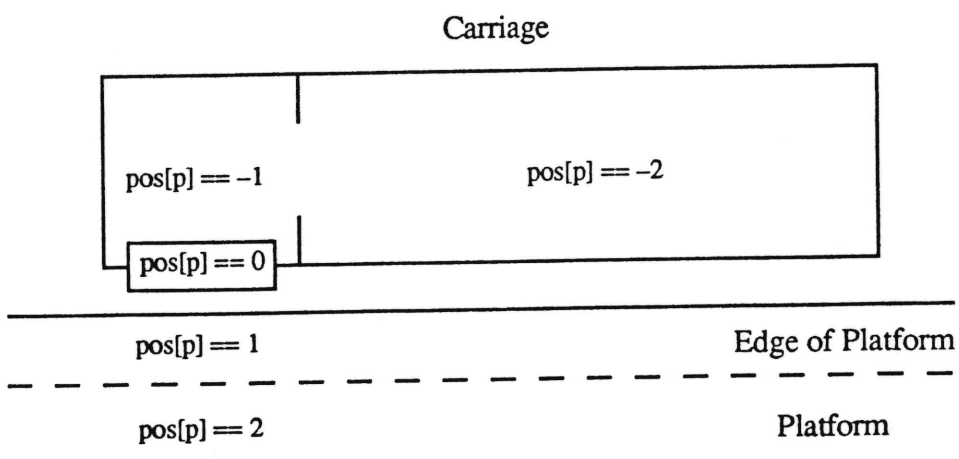

Figure 2