

VectorRally in UPPAAL

Kåre J. Kristoffersen, Henrik R. Andersen

December 8, 2003

1 Problem Description

VectorRally is a mathematical simulation of the challenges met when driving a car as fast as possible through a course with many turns and obstacles. It is natural in the sense that it involves acceleration and slowing down, and it captures quite nicely the fact that one needs to slow down before taking a turn. The purpose of this exercise is to use the model checker Uppaal to find the most ideal path (i.e. the fastest/shortest path) on a rally course.

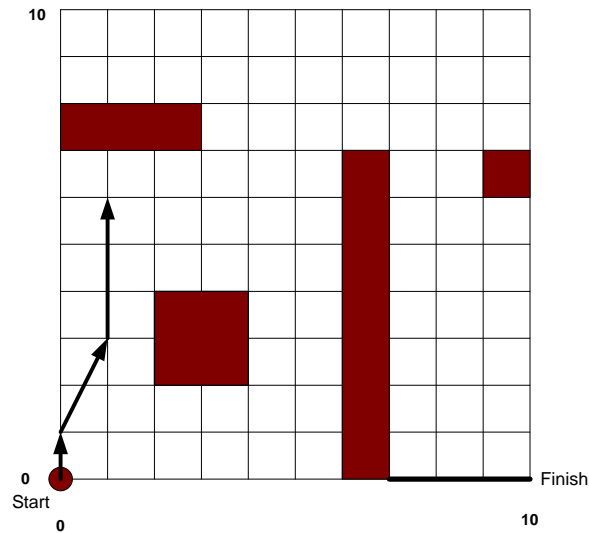


Figure 1: The car starts in position $(0,0)$ with a speed of $(0,0)$. It should reach the finish line but avoid to go out of bounds as well as hitting any obstacles, that is the car may not visit any of the obstacles neither as a vector startpoint nor as an intermediate point on the vector. Hence for instance a vector from $(5,6)$ to $(8,6)$ would be illegal.

The rally car consists of two integer vectors **position** = (x,y) and **speed** = (sx,sy) . Initially both vectors are $\vec{0}$. In each step the car changes position by adding the speedvector to the positionvector possibly by adjusting +/- 1 in horizon and vertical direction. In this way the speedvector is changed as well. See Figure 1 for an example.

2 Modelling in Uppaal

The modelling in Uppaal consist of two automata, **Car** and **Checker** as well as a two dimensional array of integers $b[4][4]$ for representing the obstacles and variables x,y,sx,sy,dx,dy .

2.1 The Car Automaton

The Car automaton proposes a new position and a change of speed and issues the synchronziation `check` to get the Checker automaton validate the move. If the checker automaton can accept the move the car can continue (synchronization on channel `ok`). Otherwise the car automaton deadlocks in state C_4 .

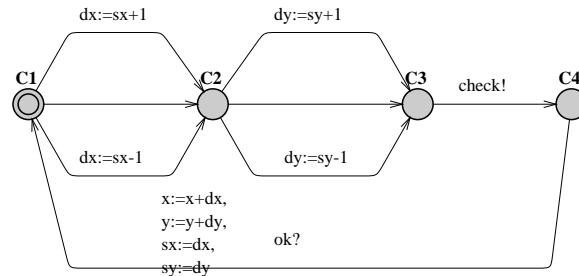


Figure 2: Car Automaton. In each iteration the car decides a change of speed horizontally and vertically and then asks the Checker to check if the move is a valid one. If this is the cas the car can continue, otherwise it deadlocks.

2.2 Shared Variables

Each of the four obstacles on the rally course are represented by lower left corner and upper right corner. In Uppaal we choose a two dimensional array of integers to represent the four obstacles. The modelling of the border of the rally course itself is described later.

```
int[0,10] b[4][4] := {{2,2,4,4},{0,7,3,8},{6,0,7,7},{9,6,10,7}};
```

The position of the car (x,y) as well as speed (sx,sy) are integers ranging from 0 to 10. The variables dx and dy are temporary variables for the speed vector.

```
int[0,10] x, y, sx, sy, dx, dy;
```

The variable `rect` denotes the rectangle currently being checked for obstruction.

```
int[0,4] rect;
```

2.3 The Checker Automaton

The key of the checker automaton is to identify if or not a vector intersects with one of the rectangles. In this section we shall describe how to detect intersection of a rectangle and a vector, see Figure 3.

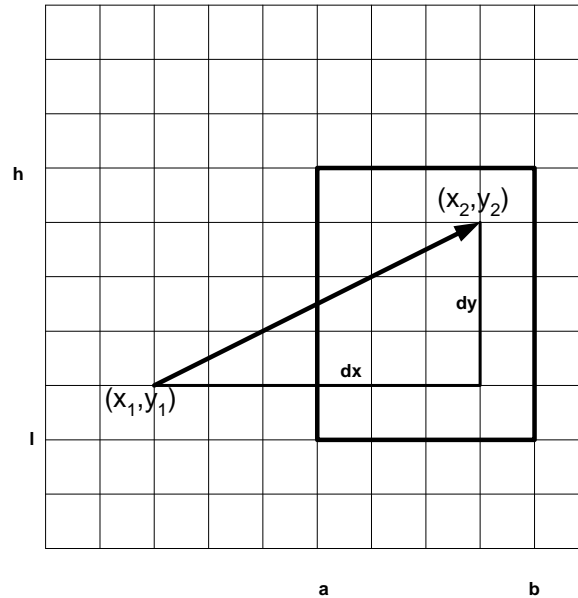


Figure 3: A rectangle with lower left corner (a, l) and upper right corner (b, h) is intersected by a vector from (x_1, y_1) to (x_2, y_2)

In the following we shall see how to detect intersection with the left side of a rectangle and a vector pointing to the right. We shall be using the fact that dx is larger than zero in this case. There will be an intersection if

$$x_1 \leq a \leq x_2 \text{ and } l \leq y_1 + (a - x_1) \frac{dy}{dx} \leq h$$

The fraction $\frac{dy}{dx}$ may be a non-integer and therefore not representable in Uppaal. However we may rewrite the rightmost expression as follows.

$$l \leq y_1 + (a - x_1) \frac{dy}{dx} \leq h \quad \Rightarrow$$

$$l \leq y_1 \cdot \frac{dx}{dx} + (a - x_1) \frac{dy}{dx} \leq h \quad \Rightarrow$$

$$l \leq \frac{y_1 \cdot dx + (a - x_1) \cdot dy}{dx} \leq h \quad \Rightarrow$$

$$l \cdot dx \leq y_1 \cdot dx + (a - x_1) \cdot dy \leq h \cdot dx$$

Crossing of the other three sides of a rectangle can be dealt with in the same manner. In Figure 4 we see the automaton Checker and in Figure 2.3 the guard on the transition from state *Ch3* to *Ch2*.

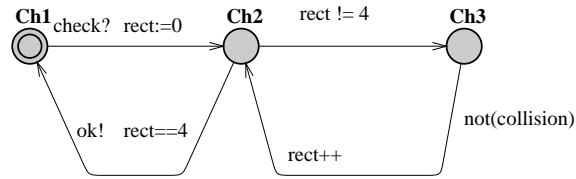


Figure 4: Checker Automaton. The checker checks for each of the four rectangles that the car does not touch the rectangle.

2.4 Result

The property of reaching the finish line in a halting position can be formulated as follows:

$$E \langle \rangle (8 \leq x \text{ and } x \leq 10 \text{ and } y == 0 \text{ and } sx == 0 \text{ and } sy == 0)$$

It takes Uppaal 1-2 seconds to to produce the answer displayed in Figure 6.

```

not(
x+dx<0 or x+dx>10 or y+dy<0 or y+dy>10 or
(dx==0 and b[rect][0]<=x and x<=b[rect][2] and
((y<=b[rect][1] and y+dy>=b[rect][1])or
(y>=b[rect][1] and y+dy<=b[rect][1])or
(y<=b[rect][3] and y+dy>=b[rect][3])or
(y>=b[rect][3] and y+dy<=b[rect][3])))
or
(dy==0 and b[rect][1]<=y and y<=b[rect][3] and
((x<=b[rect][0] and x+dx>=b[rect][0])or
(x>=b[rect][0] and x+dx<=b[rect][0])or
(x<=b[rect][2] and x+dx>=b[rect][2])or
(x>=b[rect][2] and x+dx<=b[rect][2])))
or
(((dx>0 and x<=b[rect][0] and x+dx>=b[rect][0])or(dx>0 and x>=b[rect][0]
and x+dx<=b[rect][0])) and b[rect][1]*dx<= (y*dx)+(b[rect][0]-x)*dy and
(y*dx)+(b[rect][0]-x)*dy<=b[rect][3]*dx)
or
(((dx<0 and x<=b[rect][0] and x+dx>=b[rect][0])or(dx<0 and x>=b[rect][0]
and x+dx<=b[rect][0])) and b[rect][1]*dx>= (y*dx)+(b[rect][0]-x)*dy and
(y*dx)+(b[rect][0]-x)*dy>=b[rect][3]*dx)
or
(((dx>0 and x<=b[rect][2] and x+dx>=b[rect][2])or(dx>0 and x>=b[rect][2]
and x+dx<=b[rect][2])) and b[rect][1]*dx <= (y*dx)+(b[rect][2]-x)*dy and
(y*dx)+(b[rect][2]-x)*dy<=b[rect][3]*dx)
or
(((dx<0 and x<=b[rect][2] and x+dx>=b[rect][2])or(dx<0 and x>=b[rect][2] and
x+dx<=b[rect][2])) and b[rect][1]*dx >= (y*dx)+(b[rect][2]-x)*dy and
(y*dx)+(b[rect][2]-x)*dy>=b[rect][3]*dx)
or(((dy>0 and y<=b[rect][1] and y+dy>=b[rect][1]) or (dy>0 and y>=b[rect][1]
and y+dy<=b[rect][1])) and b[rect][0]*dy<= (x*dy)+(b[rect][1]-y)*dx and
(x*dy)+(b[rect][1]-y)*dx <= b[rect][2]*dy)
or(((dy<0 and y<=b[rect][1] and y+dy>=b[rect][1]) or (dy<0 and y>=b[rect][1]
and y+dy<=b[rect][1])) and b[rect][0]*dy>= (x*dy)+(b[rect][1]-y)*dx and
(x*dy)+(b[rect][1]-y)*dx >= b[rect][2]*dy)
or(((dy>0 and y<=b[rect][3] and y+dy>=b[rect][3]) or (dy>0 and y>=b[rect][3]
and y+dy<=b[rect][3])) and b[rect][0]*dy<= (x*dy)+(b[rect][3]-y)*dx and
(x*dy)+(b[rect][3]-y)*dx <= b[rect][2]*dy)
or(((dy<0 and y<=b[rect][3] and y+dy>=b[rect][3]) or (dy<0 and y>=b[rect][3]
and y+dy<=b[rect][3])) and b[rect][0]*dy>= (x*dy)+(b[rect][3]-y)*dx and
(x*dy)+(b[rect][3]-y)*dx >= b[rect][2]*dy)
)

```

Figure 5: The guard expressing that collisions with obstacles are not allowed. Also the car is not allowed to go out of bounds.

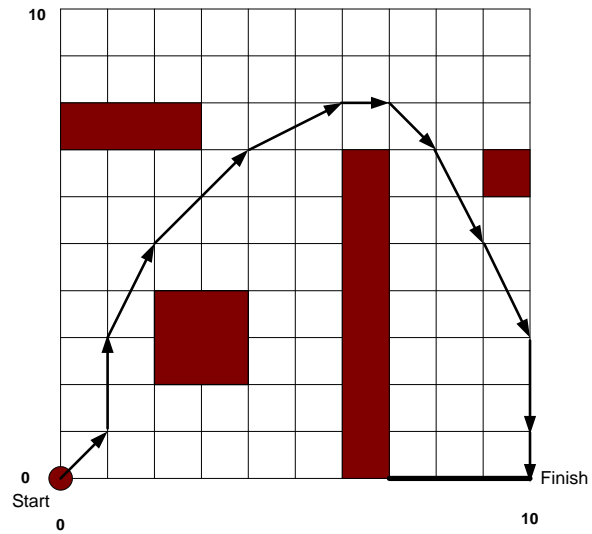


Figure 6: The shortest path from the start point to the finish line consists of 11 steps. Technically speaking there are 12 steps, the last step is used to slow down the speed from (0,-1) to (0,0) in position (10,10)