Details of the Racetrack Jani Model appearing in "Deep Statistical Model Checking"*

Timo P. Gros, Holger Hermanns, Jörg Hoffmann,
Michaela Klauck, and Marcel Steinmetz

Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
{timopgros,hermanns,hoffmann,klauck,steinmetz}@cs.uni-saarland.de

For reference, this compendium provides details regarding Racetrack and our JANI models appearing in [1]. The model, along with all other infrastructure as well as our modification of MODES is publicly available at DOI:

10.5281/zenodo.3760098 [2]

States of the vehicle are described by two vectors: its current position (x, y) indexing a cell within the grid, and its current velocity $(d_x, d_y) \in \mathbb{Z}^2$ in x and y-direction. The state of the vehicle is updated at discrete steps. At each step, the speed of the vehicle can be controlled via 9 different actions corresponding to the acceleration vectors $(a_x, a_y) \in (\{-1, 0, 1\})^2$. Acceleration is applied additively, i.e., the vehicle's new velocity vector (d'_x, d'_y) after applying acceleration (a_x, a_y) is given by $d'_x = d_x + a_x$ and $d'_y = d_y + a_y$. The position of the vehicle is updated according to the updated velocity vector, i.e., $x' = x + d'_x$ and $y' = y + d'_y$.

What we just specified is the deterministic variant of Racetrack. In the noisy variant, acceleration only succeeds with a probability of $p \in [0, 1)$, while with probability $(1 - p)$ the vehicle's velocity remains the same.

We say that the vehicle has *crashed* if the vehicle either moved out of the grid (i.e., its position no longer constitutes a valid grid coordinate), or the vehicle's last movement trajectory crossed a wall cell. Determining whether the vehicle has crashed is done by discretizing the trajectory from the vehicle's former position $(x_0, y_0) := (x, y)$ to its new position $(x_n, y_n) := (x', y')$ into a sequence of coordinates $T = \langle (x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n) \rangle$. Then, the vehicle has touched a wall iff T references a coordinate of a wall cell. Checking whether the vehicle traversed a goal cell is done in the same fashion. The trajectory discretization T is defined as follows:

$$T = \begin{cases} \langle (x,y) \rangle & \text{if } d_x = 0 \text{ and } d_y = 0 \quad (1) \\ \langle (x,y), (x+\sigma_x, y), (x+2\cdot\sigma_x, y) \ldots, (x', y') \rangle & \text{if } d_x \neq 0 \text{ and } d_y = 0 \quad (2) \\ \langle (x,y), (x, y+\sigma_y), (x, y+2\cdot\sigma_y) \ldots, (x', y') \rangle & \text{if } d_x = 0 \text{ and } d_y \neq 0 \quad (3) \\ \langle (x,y), (x+\sigma_x, \lfloor y + m_y \rceil), (x+2\cdot\sigma_x, \lfloor y + 2\cdot m_y \rceil) \ldots, (x', y') \rangle & \text{if } d_x \neq 0 \text{ and } d_y \neq 0 \text{ and } |d_x| \geq |d_y| \quad (4) \\ \langle (x,y), (\lfloor x + m_x \rceil, y+\sigma_y), (\lfloor x + 2\cdot m_x \rceil, y+2\cdot\sigma_y) \ldots, (x', y') \rangle & \text{if } d_x \neq 0 \text{ and } d_y \neq 0 \text{ and } |d_x| < |d_y| \quad (5) \end{cases}$$

where $\sigma_x = sgn(d_x)$, $\sigma_y = sgn(d_y)$ and $m_x = \frac{d_x}{|d_y|}$, $m_y = \frac{d_y}{|d_x|}$. In words, if either the horizontal or vertical speed is 0 (cases 1 to 3), the trajectory contains exactly all grid coordinates on the straight line between (x, y) and (x', y'). Otherwise, we linearly interpolate n points between the two positions and then for each such point round to

* Authors are listed alphabetically.

the closest position on the map. In our model n is given by $\max(|d_x|, |d_y|)$, while the original discretization model always choose $n = d_x$. The latter is problematic when having a velocity which moves more into the y (case 5) than into the x direction (case 4), as then only few point will be contained in the trajectory and counterintuitive results are produced.

Our JANI implementation is a straightforward encoding of the model described above. The race track is encoded as a (constant) two-dimensional array whose size equals that of the grid. The JANI files of different Racetrack instances differ only in this array.

Vehicle movements and collision checks are represented by separate automata that synchronize using shared actions. The vehicle automaton keeps track of the current state of the vehicle via four bounded integer variables (position and directional velocity), and two Boolean variables (indicating whether the vehicle has crashed, or has reached a goal cell). The automaton starts at a location with one edge for each one of the 9 different acceleration vectors. Each of the edges updates the velocity accordingly, and sends the start and resulting end coordinates to the collision check automaton. The collision check can respond with three different answers: "valid", "crash", or reached "goal". If the trajectory was valid, the vehicle automaton transitions back to its initial location. Otherwise the vehicle automaton transitions into a terminal location where no further moves are possible.

The collision check automaton takes care of two things. It first checks whether the vehicle's destination lies within the grid. If so, it then iteratively computes the discretized trajectory T, and looks up for each referenced coordinate whether the corresponding entry in the grid array represents a wall or goal cell. If the trajectory leads out of the track, or when an intersection of the trajectory with either a wall or a goal cell is detected, the result is immediately sent to the vehicle automaton. If the trajectory was completely generated without detecting a collision, the vehicle automaton's request is answered with "valid", and the location is reset, waiting for the next trajectory to test.

References

1. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Steinmetz, M.: Deep Statistical Model Checking. In: Proceedings of the 40th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE'20) (2020), available at https://doi.org/10.1007/978-3-030-50086-3_6
2. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Steinmetz, M.: Models and Infrastructure used in "Deep Statistical Model Checking" (2020), available at http://doi.org/10.5281/zenodo.3760098