CriSGen: Constraint-based Generation of Critical Scenarios for Autonomous Vehicles

Andreas Nonnengart, Matthias Klusch, and Christian Müller

German Research Center for Artificial Intelligence, Saarbrücken, Germany
{firstname.lastname}@dfki.de

Abstract. Ensuring pedestrian-safety is paramount to the acceptance and success of autonomous cars. The scenario-based training and testing of such self-driving vehicles in virtual driving simulation environments has increasingly gained attention in the past years. A key challenge is the automated generation of critical traffic scenarios which usually are rare in real-world traffic, while computing and testing all possible scenarios is infeasible in practice. In this paper, we present a formal method-based approach **CriSGen** for an automated and complete generation of critical traffic scenarios for virtual training of self-driving cars. These scenarios are determined as close variants of given but uncritical and formally abstracted scenarios via reasoning on their non-linear arithmetic constraint formulas, such that the original maneuver of the self-driving car in them will not be pedestrian-safe anymore, enforcing it to further adapt the behavior during training.

Keywords: Autonomous Driving · Formal Methods · Critical Scenarios.

1 Introduction

Scenario-based training and testing of autonomous vehicles in driving simulators gained quite some attention recently. In fact, from an ethical perspective, synthesizing critical traffic scenarios in order to virtually train self-driving cars to perform pedestrian-safe (pedestrian collision avoiding) navigation suggests itself. Such scenarios are usually rare in real-world traffic and computing all possible scenarios for extracting unsafe ones is infeasible in practice. This challenge is addressed by various approaches to automated traffic scenario generation based on formal methods [1, 8, 4, 13] or evolutionary and deep learning methods [19, 14]. However, none of these generation approaches take known safe maneuvers of the self-driving car in given scenarios into account in order to determine critical traffic scenarios.

To this end, we developed a novel formal method-based approach **CriSGen** for an automated, complete generation of critical traffic scenarios for virtual training of self-driving cars. Critical scenarios are determined as close variants of given but uncritical maneuver and scenario abstraction via formal reasoning on non-linear arithmetic constraint formulas with free parameters, such that

the original maneuver of the self-driving car in them will not be pedestrian-safe anymore.

The remainder of the paper is structured as follows. A brief overview of the approach is given in Section 2, while its formal analysis techniques are described in more detail in Section 3. An illustrative example is provided in Section 4, and related work is summarized in Section 5 before we conclude in Section 6.

2 CriSGen Overview

The overall approach of **CriSGen** is illustrated in Figure 1. An autonomous car operates in a virtual driving simulation environment such as OpenDS [5] and utilizes some learning technique for pedestrian-safe maneuver training. In each simulated traffic scenario, the adaptive car control determines its maneuver actions in terms of acceleration and steering, and is supposed to update its action policy based on feedback from the simulation environment such as whether it nearly misses or even hits a pedestrian. The automated generation of criti-

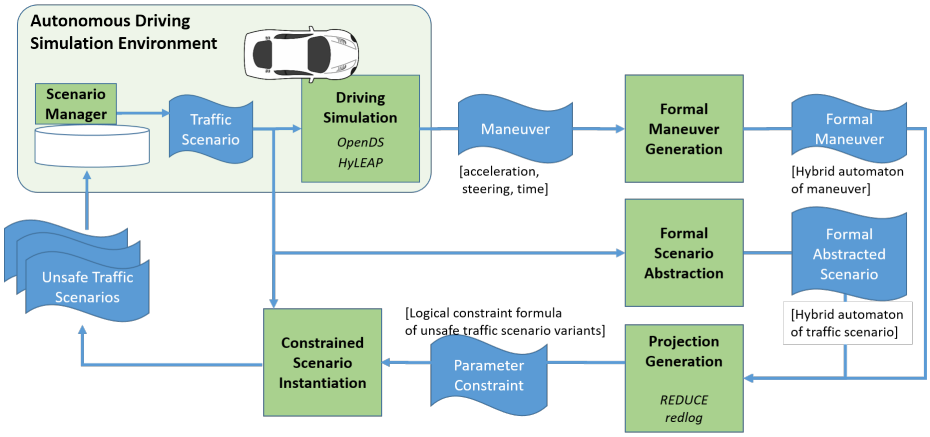

Fig. 1. Overview of the CriSGen approach for critical scene generation

cal traffic scenarios by **CriSGen** can be triggered at any time, in particular in cases where the self-driving car appears to behave pedestrian-safe for some time period. The **CriSGen** process starts with transforming both the car maneuver action in the considered (uncritical) traffic scenario into a formal model and the scenario itself into a formal abstraction by replacing some of its concrete values with free parameters such that it represents a whole range of variants of the original scenario. These formal models together with a suitable unsafe-property (cf. Sect. 3.1) are then automatically analyzed (cf. Sect. 3.2) to obtain a non-linear arithmetic constraint formula which reflects all those scenario variants

that must be considered *critical* for the given maneuver of the car. Geometrically, what we obtain this way is a collection of regions in n-dimensional space (where n is the number of free parameters) such that each point in any of these regions represents a valuation of the parameters that makes the maneuver unsafe. The original scenario is represented as a single point in this n-dimensional space and lies outside any of the unsafe regions. **CriSGen** then selects points in the unsafe regions that are as close as possible, i.e. geometrically near to the original scenario (cf. Sect. 3.3). In other words, we translate and compose the abstracted scenario together with the original maneuver into some sort of hybrid automaton. A forward reachability analysis with respect to some suitable unsafety property ("hitting pedestrians") ends up in a constraint formula whose solutions are exactly the representatives of critical scenarios. In the final instantiation steps, a whole bunch of critical scenarios are obtained which are returned to the scenario manager of the driving simulator for challenging the car.

Our implementation setting for **CriSGen** employs OpenDS [5] as a virtual driving simulator and HyLEAP [17] for adaptive maneuver training of the car in OpenDS. For reachability analysis one might adapt systems like PVS [12] or TLA+ [15]. Instead, **CriSGen** makes utilizes the general-purpose computer algebra system REDUCE, in particular, its module **redlog** [6], to perform quantifier elimination on non-linear arithmetic constraint formulas. For instance, given that $a > 0$, the quantifier-free equivalent of $\exists x\ ax^2 + bx + c \leq 0$ is $b^2 \geq 4ac$ [7]; **redlog** determines the projections of complex constraint formulas onto the set of variables of interest, for example the parameters introduced by traffic scenario abstractions. In the remainder of this paper, whenever we speak of the quantifier-free equivalent of a (quantified) constraint formula we mean **redlog**'s quantifier elimination output when called with this constraint formula.

3 CriSGen: Formal Models and Analysis

3.1 Formal Models of Maneuver and Scenario

The formal analyses by **CriSGen** require formal models of car maneuvers, (abstracted) traffic scenarios, the composition of these models and an unsafe-property to be analyzed. In the following, we assume traffic scenarios with a single pedestrian, though the approach is not restricted to that.

Abstracted Scenario. We assume a two-dimensional grid upon which the pedestrian and the autonomous vehicle move. Each point on this grid is called a *position*, and the movements of pedestrians are described in sequences of positions together with a scalar velocity as[1]

$$(x_0, y_0) \xrightarrow{v_0} (x_1, y_1) \longrightarrow \ldots \longrightarrow (x_i, y_i) \xrightarrow{v_i} (x_{i+1}, y_{i+1}) \longrightarrow \ldots$$

In the original scenario all the x's, y's, and v's are concrete numbers, while in the next step we relax this and produce abstracted variants of this scenario with

[1] OpenDS scenarios are described in specific XML files from which such behaviors can be extracted.

the help of what we called *Abstraction Modifiers*. These are operators that can be applied to a given scenario (that may be already partially abstracted):

1. Replace a waypoint component with a (fresh) parameter (with or without propagation)
2. Replace a segment velocity with a (fresh) parameter
3. Split a segment (thus adding a waypoint and a parameter)
4. Double a waypoint

The application of an Abstraction Modifier has an instance, i. e. an instantiation of the free parameters, that is behavior equivalent to the original behavior. Regarding propagation, let us consider the simple scenario $(0,0) \xrightarrow{1} (0,1) \xrightarrow{1} (10,1)$, and suppose that we want to replace the y-component of the second waypoint with parameter c. Without propagation this ends up with the result $(0,0) \xrightarrow{1} (0,c) \xrightarrow{1} (10,1)$ as one might have expected. *With* propagation, however, would consider the y-component of the third waypoint as a function of the abstracted one. In the example this means that the two y-components should remain equal, i. e. we end up with $(0,0) \xrightarrow{1} (0,c) \xrightarrow{1} (10,c)$.

Splitting of a segment can be formulated in terms of quantifier elimination. For instance, to split a segment $(1,2) \xrightarrow{1} (2,5)$ we determine the quantifier-free equivalent of $\exists\lambda\ (0 \leq \lambda \leq 1 \wedge a = 1+\lambda \wedge b = 2+3\lambda)$ which is $1 \leq a \leq 2 \wedge b = 3a-1$. The split segment therefore is $(1,2) \xrightarrow{1} (a, 3a-1) \xrightarrow{1} (2,5)$, where $1 \leq a \leq 2$. Of course, this also works in cases where parameters have already been introduced. For example, splitting the segment $(1,2) \xrightarrow{1} (3,p)$, where p is a parameter, results in the split segment $(1,2) \xrightarrow{1} (a, \frac{1}{2}(ap - 2a - p + 6)) \xrightarrow{1} (3,p)$ with $1 \leq a \leq 3$. A scenario modification is then defined as the successive application of several such Abstraction Modifiers.

Formal Model of Abstracted Scenario. Let (p_x, p_y) denote the pedestrian's position (x and y component) with both values being functions over time, such that the respective velocity components are the first derivatives of p_x and p_y denoted by $\dot{p}_x$ and $\dot{p}_y$. These components have to be specified in order to be able to describe the reachable positions of the pedestrian within a phase of the scenario. Since the pedestrian walks from (x_i, y_i) to (x_{i+1}, y_{i+1}) with velocity v_i (a scalar), we have that $v_i^2 = \dot{p}_x^2 + \dot{p}_y^2$. Besides, the pedestrian takes the same time to cross the distance $x_{i+1} - x_i$ as the distance $y_{i+1} - y_i$: the ratio $(x_{i+1} - x_i)/(y_{i+1} - y_i)$ is the same as the ratio $\dot{p}_x/\dot{p}_y$, and so $\dot{p}_y\ (x_{i+1} - x_i) = \dot{p}_x\ (y_{i+1} - y_i)$ holds. This does not yet uniquely describe the velocity components. In order to make sure that the velocities have the correct sign we also add $\dot{p}_x\ (x_{i+1} - x_i) \geq 0$ and $\dot{p}_y\ (y_{i+1} - y_i) \geq 0$. Together, all these (in)equations fully describe the pedestrian's continuous dynamics.

Next, we have to make sure that the pedestrian completes the current segment as soon as she reaches (x_{i+1}, y_{i+1}), and that she passes through each point of the line segment while walking. This gives rise to the following segment invariant: If $x_{i+1} \geq x_i$ it suffices to add the invariant $p_x \leq x_{i+1}$ (and

analogously for the y-component). Similarly, if $x_{i+1} \leq x_i$ then the invariant $p_x \geq x_{i+1}$ would do. However, we can avoid such a case distinction if we declare $p_x\ (x_{i+1} - x_i) \leq x_{i+1}\ (x_{i+1} - x_i)$ and $p_y\ (y_{i+1} - y_i) \leq y_{i+1}\ (y_{i+1} - y_i)$ as our phase invariant. Obviously, if the distance is positive then the differences cancel out and if it is negative they cancel out but also reverse the inequality sign. The two lower implications capture the marginal cases where the pedestrian moves straight in the y-direction (x-direction respectively).

Definition 1 (Specification of pedestrian dynamics and invariant for segment i). *Let segment i of an abstracted scenario be $(x_i, y_i) \xrightarrow{v_i} (x_{i+1}, y_{i+1})$. Let $\Delta x = x_{i+1} - x_i$ and $\Delta y = y_{i+1} - y_i$ We define pedestrian p dynamics dyn_i^p and invariant inv_i^p for segment i as follows:*

$$\mathtt{dyn}_i^p = \begin{bmatrix} v_i^2 = \dot{p}_x^2 + \dot{p}_y^2 \wedge v_i \geq 0 & \wedge \\ \dot{p}_x\ \Delta x \geq 0 & \wedge \\ \dot{p}_y\ \Delta y \geq 0 & \wedge \\ \Delta x\ \dot{p}_y = \Delta y\ \dot{p}_x & \end{bmatrix} \quad \mathtt{inv}_i^p = \begin{bmatrix} p'_x\ \Delta x \leq x_{i+1}\ \Delta x & \wedge \\ p'_y\ \Delta y \leq y_{i+1}\ \Delta y & \wedge \\ x_{i+1} = x_i \rightarrow p'_x = p_x & \wedge \\ y_{i+1} = y_i \rightarrow p'_y = p_y & \end{bmatrix}$$

With these definitions we can imagine an (hybrid) automaton-like representation of the translation of an abstracted scenario as a sequence of nodes each representing one segment Seg_i with continuous dynamics $\mathtt{dyn}_i^p$, invariant $\mathtt{inv}_i^p$ and transition guards $p_x = x_{i+1}, p_y = y_{i+1}$. Informally, the reachability semantics defines the set of reachable states, in this case the pedestrian's positions, for each of the segments (cf. Sect. 3.2).

Formal Model of Maneuver. The virtually simulated autonomous car outputs a maneuver description that starts with an initial position (α, β) together with an initial x, y-velocity (μ_0, ν_0) followed by a sequence of maneuver events each accompanied with a τ that expresses the duration of the current state of movement. A typical maneuver would thus be:

$$((\alpha, \beta), (\mu_0, \nu_0)) \xrightarrow{\tau_0} \mathtt{e_1(n_1)} \xrightarrow{\tau_1} \mathtt{e_2(n_2)} \xrightarrow{\tau_2} \mathtt{e_3(n_3)} \xrightarrow{\tau_3} \dots$$

where each $\mathtt{e_i(n_i)}$ denotes one of the maneuver events from below

`de-/accelerate by` n: De-/Increase velocity while keeping the direction
`steer left/right by` ϕ: ϕ might be in degrees or radians, velocity remains constant

In the course of driving these car dynamics change from (μ_j, ν_j) to (μ_{j+1}, ν_{j+1}) depending on the current maneuver event. Thus, for modelling the maneuver, we determine the – as we call it – velocity sequence as follows:

$$(\mu_0, \nu_0), (\mu_1, \nu_1), (\mu_2, \nu_2), (\mu_3, \nu_3), \dots$$

(μ_0, ν_0) is already given in the maneuver description. Having (μ_j, ν_j), the next velocity vector (μ_{j+1}, ν_{j+1}) depends on the maneuver event $\mathtt{e_{j+1}(n_{j+1})}$. In case of a steering event, we simply multiply by the rotation matrix, i e.,

$$\begin{pmatrix} \mu_{j+1} \\ \nu_{j+1} \end{pmatrix} = \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} \mu_j \\ \nu_j \end{pmatrix}$$

Otherwise, $\mathtt{e_{j+1}}(\mathtt{n_{j+1}})$ is a de-/acceleration event. In this case, (μ_{j+1}, ν_{j+1}) is uniquely characterized by the quantifier-free equivalent of

$$\exists\, \mathtt{old}, \mathtt{new} \quad \begin{bmatrix} \mathtt{old}^2 = \mu_j^2 + \nu_j^2 \wedge \mathtt{old} \geq 0 \quad \wedge \\ \mathtt{new}^2 = \mu_{j+1}^2 + \nu_{j+1}^2 \wedge \mathtt{new} = \mathtt{old} + \mathtt{n_{j+1}} \quad \wedge \\ \mu_j\ \nu_{j+1} = \mu_{j+1}\ \nu_j \quad \wedge \\ \mu_j\ \mu_{j+1} \geq 0 \wedge \nu_j\ \nu_{j+1} \geq 0 \end{bmatrix}$$

The top four in-/equations express the change in velocity, the fifth guarantees that the absolute value of the direction is kept constant, and the final two equations make sure that the velocities are not reversed[2]. With these definitions the characterization of the car's continuous dynamics and invariant for phase j – in the sequel denoted by Ph_j – becomes

$$\mathtt{dyn}_j^c = \left[\, \dot{c}_x = \mu_j \wedge \dot{c}_y = \nu_j \wedge \dot{t} = 1 \,\right] \qquad \mathtt{inv}_j^c = \left[\, t' \leq \tau_j \,\right]$$

In terms of hybrid automata, we obtain again a sequence of nodes, each responsible for a maneuver phase with continuous dynamics $\mathtt{dyn}_j^c$, invariant $\mathtt{inv}_j^c$ and transition annotations $t = \tau_j, t' = 0$. The formal model of a maneuver in terms of hybrid automata (with reachable state semantics defined in Sect. 3.2) can therefore be described as in Figure 2.

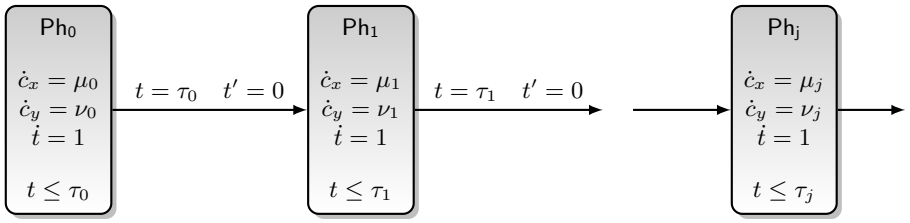

Fig. 2. Hybrid automaton for maneuvers.

Composition of Formal Models of Scenario and Maneuver. Given an abstracted scenario $(x_0, y_0) \xrightarrow{v_0} (x_1, y_1) \xrightarrow{v_1} \ldots$ and a maneuver $((\alpha, \beta), (\mu_0, \nu_0)) \xrightarrow{\tau_0} \mathtt{e_1}(\mathtt{n_1}) \xrightarrow{v_0} \ldots$ with velocity sequence $(\mu_0, \nu_0), (\mu_1, \nu_1), \ldots$, we obtain the pedestrian's (hybrid) automaton and the car's maneuver (hybrid) automaton as described above. There are no synchronization labels involved, nor are there any cycles. The composition of these two hybrid automata is therefore straightforward: Its nodes are the cross-products $(Seg_i \times Ph_j)$ of the scenario and the maneuver nodes, the continuous dynamics and the invariants are the conjunction of the local dynamics and invariants, and the transitions are the local transitions from the local automata, i. e. two outgoing transitions for each node.

[2] This disallows maneuvers in which a car drives slowly by, say, 1 m/s, and decelerates by 2 m/s, thus reversing its motion direction. Reversing the sense of direction should be described by decelerating to a stop and a further deceleration for driving backwards.

Safety Property. Pedestrian avoidance serves as a unique safety property to be satisfied by the self-driving car. To this end, the pedestrian hit area of the car is defined in terms of the car's position and direction. A maneuver is called **safe** if there is no point in time where a pedestrian's position lies within the car's hit area. Let the car's position and velocity vector be (c_x, c_y) and $(\dot{c}_x, \dot{c}_y)$, respectively. One vector perpendicular to the sense of direction is $(\dot{c}_y, -\dot{c}_x)$ and the car's speed is $\sqrt{\dot{c}_x^2 + \dot{c}_y^2}$, which we abbreviate to α. For checking the safety of the pedestrian's position, we extend the car's position (c_x, c_y) by at most ± 3 (meters, say)[3] in the sense of direction, which is normalized to $(\dot{c}_x, \dot{c}_y)/\alpha$, and at most ± 1 perpendicular to the sense of direction, which is normalized to $(\dot{c}_y, -\dot{c}_x)/\alpha$. After minor simplifications, we finally end up with: The position (p_x, p_y) is inside the car's hit area (during node $Seg_i \times Ph_j$) iff $(p_x, p_y, c_x, c_y) \in \mathtt{Hit}_{(i \times j)}$ where

$$\mathtt{Hit}_{(i \times j)} = \exists \lambda_1, \lambda_2, \alpha \begin{bmatrix} \alpha^2 = \dot{c}_x^2 + \dot{c}_y^2 \ \wedge \ \alpha \geq 0 \quad \wedge \\ -1 \leq \lambda_1 \leq 1 \ \wedge \ -3 \leq \lambda_2 \leq 3 \quad \wedge \\ \alpha \ c_x - \alpha \ p_x = \lambda_1 \ \dot{c}_y + \lambda_2 \ \dot{c}_x \quad \wedge \\ \alpha \ c_y - \alpha \ p_y = \lambda_2 \ \dot{c}_y - \lambda_1 \ \dot{c}_x \end{bmatrix}$$

with $\dot{c}_x = \mu_j$ and $\dot{c}_y = \nu_j$. Obviously, this safety property changes from phase to phase of the composed automaton, since velocities and directions of participants vary from phase to phase, and the safety property depends on these values.

3.2 Formal Analysis of Abstracted Scenarios and Maneuvers

For the formal assessment of given maneuvers with respect to abstracted scenarios we define what we understand by the set of reachable states and how these are represented. Therefore let us assume that we have an abstracted scenario

$$(x_0, y_0) \xrightarrow{v_0} (x_1, y_1) \xrightarrow{v_1} \ldots \xrightarrow{v_{i-1}} (x_i, y_i) \xrightarrow{v_i} (x_{i+1}, y_{i+1}) \xrightarrow{v_{i+1}} \ldots$$

and a maneuver

$$((\alpha, \beta), (\mu, \nu)) \xrightarrow{\tau_0} \mathtt{e_1}(\mathtt{n_1}) \xrightarrow{\tau_1} \mathtt{e_2}(\mathtt{n_2}) \xrightarrow{\tau_2} \mathtt{e_3}(\mathtt{n_3}) \xrightarrow{\tau_3} \ldots$$

Since the continuous dynamics change from node to node in the composition we consider the set of reachable states as the union of the sets of reachable states for the various composed nodes. In each such node the respective velocities are considered constant, therefore let

$$\mathtt{Reach} = \begin{bmatrix} \delta \geq 0 \wedge t' = t + \delta \quad \wedge \\ p_x' = p_x + \delta \ \dot{p}_x \wedge p_y' = p_y + \delta \ \dot{p}_y \quad \wedge \\ c_x' = c_x + \delta \ \dot{c}_x \wedge c_y' = c_y + \delta \ \dot{c}_y \end{bmatrix}$$

This together with the local continuous dynamics and invariant gives rise to the following definition of the set of reachable states.

[3] The units do not really matter as long as they are kept consistent throughout the specification.

Definition 2 (Reachable States for Node $Seg_i \times Ph_j$).
Let $vars = \{p_x, p_y, \dot{p}_x, \dot{p}_y, c_x, c_y, \dot{c}_x, \dot{c}_y, t, \delta\}$. Then the set of reachable states in node $Seg_i \times Ph_j$, $\texttt{States}_{(i\times j)}$, is uniquely determined by (the quantifier-free equivalent of)

$$\texttt{States}_{(i\times j)} = \exists\, vars\ \left[\texttt{Init}_{(i\times j)} \wedge \texttt{Dyn}_{(i\times j)} \wedge \texttt{Reach} \wedge \texttt{Inv}_{(i\times j)}\right]$$

where $\texttt{Dyn}_{(i\times j)} = dyn_i^p \wedge dyn_j^c$ and $\texttt{Inv}_{(i\times j)} = inv_i^p \wedge inv_j^c$. For $\texttt{Init}_{(i\times j)}$ see below.

In fact, according to the definitions of $\texttt{Dyn}_{(i\times j)}$, $\texttt{Inv}_{(i\times j)}$, and $\texttt{Reach}$ the constraint $\texttt{States}_{(i\times j)}$ talks about parameters and primed variables only. For convenience, we rename these variables to their unprimed versions[4].

$$\texttt{States}_{(i\times j)} = \texttt{States}_{(i\times j)}[p_x/p'_x][p_y/p'_y][c_x/c'_x][c_y/c'_y][t/t']$$

The above definition requires $\texttt{Init}_{(i\times j)}$, the constraint that describes the initial states for node $Seg_i \times Ph_j$. These depend on the reachable states of "earlier" nodes where the transition guards hold.

Definition 3 (Initial States for Node $Seg_i \times Ph_j$). *Given an abstracted scenario $(x_0, y_0) \xrightarrow{v_0} (x_1, y_1) \xrightarrow{v_1} \ldots \xrightarrow{v_{i-1}} (x_i, y_i) \xrightarrow{v_i} (x_{i+1}, y_{i+1}) \xrightarrow{v_{i+1}} \ldots$ and a maneuver $((\alpha, \beta), (\mu, \nu)) \xrightarrow{\tau_0} \mathtt{e_1}(\mathtt{n_1}) \xrightarrow{\tau_1} \mathtt{e_2}(\mathtt{n_2}) \xrightarrow{\tau_2} \mathtt{e_3}(\mathtt{n_3}) \xrightarrow{\tau_3} \ldots$ with derived velocity sequence $(\mu_0, \nu_0), (\mu_1, \nu_1), (\mu_2, \nu_2), (\mu_3, \nu_3), \ldots$ we define*

$$\texttt{Init}_{(0\times 0)} = [p_x = x_0 \wedge p_y = y_0 \wedge c_x = \alpha \wedge c_y = \beta \wedge t = 0]$$

$$\texttt{Init}_{(0\times(j+1))} = \left[\exists t\, \{\texttt{States}_{(0\times j)} \wedge t = \tau_j\} \wedge t = 0\right]$$

$$\texttt{Init}_{((i+1)\times 0)} = \left[\texttt{States}_{(i\times 0)} \wedge p_x = x_{i+1} \wedge p_y = y_{i+1}\right]$$

$$\texttt{Init}_{((i+1)\times(j+1))} = \begin{bmatrix} \texttt{States}_{(i\times(j+1))} \wedge p_x = x_{i+1} \wedge p_y = y_{i+1} \\ \vee \\ \exists t\, \{\texttt{States}_{((i+1)\times j)} \wedge t = \tau_j\} \wedge t = 0 \end{bmatrix}$$

Finally, after having determined the reachable states (for node $Seg_i \times Ph_j$) and having found the hit area (also for node $Seg_i \times Ph_j$), the constraint describing unsafe states is as follows.

Definition 4 (Unsafe States for Node $Seg_i \times Ph_j$). *Given $\texttt{States}_{(i\times j)}$ and $\texttt{Hit}_{(i\times j)}$, the unsafe states for node $Seg_i \times Ph_j$ are defined as the quantifier-free equivalent of*

$$\texttt{Unsafe}_{(i\times j)} = \exists p_x, p_y, c_x, c_y, t\ \left[\texttt{States}_{(i\times j)} \wedge \texttt{Hit}_{(i\times j)}\right]$$

Each of the `Unsafe`-constraints contains no variable at all, and not all of them are simply `true` or `false`. In general, they still contain constraints over the parameters that had been introduced by Abstraction Modifiers. Algorithm 3.1 summarizes the forward-reachability mechanism defined above. Note that the nested for-loop guarantees that the constraint predicates are determined just-in-time.

[4] This can trivially be described as a quantifier elimination problem.

Algorithm 3.1: ForwardReachability($Abstr.Scenario, Maneuver$)

```
Critical ← false
for i ← 0, 1, 2, ...
do { for j ← 0 to i
     do { Init[j][i − j] ← see Definition 3
          Dyn[j][i − j] ← see Definition 1
          Inv[j][i − j] ← see Definition 1
          States[j][i − j] ← see Definition 2
          Hit[j][i − j] ← see Section 3.1
          Unsafe[j][i − j] ← see Definition 4
          Critical ← Critical ∨ Unsafe[j][i − j] }
return (Critical)
```

3.3 Generating Critical Scenarios

Suppose that the scenario abstraction introduced the parameters $\{p_1, \ldots, p_n\}$. Now consider an n-dimensional grid with axes $p_1, \ldots, p_n$. Each point in this grid represents a variant of the abstracted scenario. The closer two such points are, the more similar are the variants that they represent. Some of these variants are behavior equivalent to the original scenario by definition (of the Abstraction Modifiers). Let O denote the area within this grid of the variants that are behavior equivalent to the original scenario. Algorithm 3.1 provides us with the constraint `Critical` that describes the variants that are unsafe with respect to the original maneuver. We determine the distance between the areas O and `Critical` by solving an optimization problem along the lines of [7]: The distance between any point $(a_1, \ldots, a_n)$ that satisfies O and any other point $(b_1, \ldots, b_n)$ that satisfies `Critical` is greater than or equal to the minimal distance between the two areas. Thus the quantifier-free equivalent of

$$\forall a_1, \ldots, a_n, b_1, \ldots b_n \; O[a_i/p_i] \wedge \texttt{Critical}[b_i/p_i] \rightarrow \sqrt{\sum_{i=1}^{n}(a_i^2 - b_i^2)} \geq d$$

is some constraint $d \leq \texttt{min}$ from which we can read `min` as the minimal distance that we are interested in. With the computation of witnesses for the e_i in

$$\exists c_1, \ldots, c_n, e_1, \ldots e_n \; O[c_i/p_i] \wedge \texttt{Critical}[e_i/p_i] \wedge \texttt{min} \leq \sqrt{\sum_{i=1}^{n}(c_i^2 - e_i^2)} \leq \texttt{min} + \epsilon$$

(where ϵ is a small non-negative constant[5]) we have finally found the most appropriate candidates for instantiation.

[5] ϵ compensates minor differences between the computed reachable states and the driving simulator's behavior.

4 An Illustrative Example

Scenario and Maneuver. As an illustrative example of our formal method-based approach to the generation of critical scenarios suppose that the simulation engine provides **CriSGen** with the traffic scenario and maneuver as depicted in Figure 3. In this scenario, a pedestrian starts with velocity 1 (m/sec, say) at

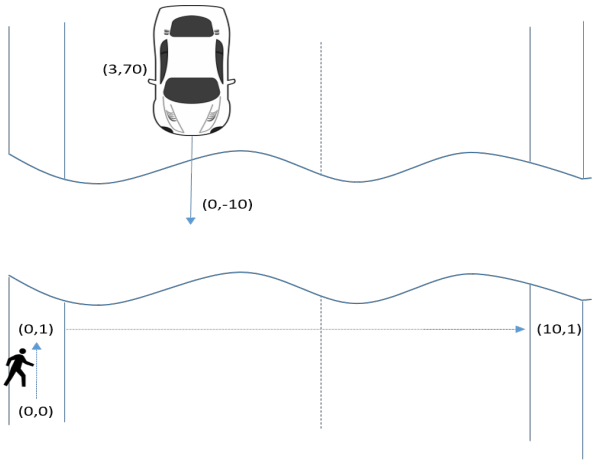

Fig. 3. Original (non-critical) scenario: The autonomous car drives too slow to jeopardize the pedestrian.

position $(0,0)$ goes straight to $(0,1)$, and crosses the street for point $(10,1)$. The car's maneuver simultaneously starts at $(3,70)$ while driving downwards with velocity 0 in x-direction and velocity -10 in y-direction. Both scenario and maneuver is summarized as follows:

$$(0,0) \xrightarrow{1} (0,1) \xrightarrow{1} (10,1) \qquad ((3,70),(0,-10)) \xrightarrow{\infty}$$

One-Dimensional Abstraction. Consider a single abstraction, namely replacing the y-component of the second waypoint by a parameter c (unrestricted) *with propagation*, which ends up in the abstracted scenario

$$(0,0) \xrightarrow{1} (0,c) \xrightarrow{1} (10,c)$$

By applying Algorithm 3.1 to this abstraction, we obtain two `Unsafe`-constraints: $\texttt{Unsafe}_{(0\times 0)} = \texttt{false}$ as expected; the pedestrian's behavior in the initial phase is certainly not critical (yet). The other `Unsafe`-constraint is more interesting:

$$\texttt{Unsafe}_{(1\times 0)} = 27 \leq 11c \leq 53 \ \vee \ -53 \leq 9c \leq -27$$

Accordingly, the critical region consists of two parts: One where the pedestrian walks towards the car for some distance and then crosses the street, and another

one, where the pedestrian actually walks away from the car before crossing the street (see Figure 4). The ultimate goal of **CriSGen** is to synthesize critical scenarios that are possibly near the (non-critical) original traffic scenario (the circle at position 1 in Figure 4). For this example, values 2.5 and -3.5 are reasonable.

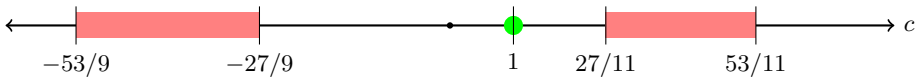

Fig. 4. Unsafe (red rectangles) and original (green circle) scenarios.

As a consequence, in this simple example, **CriSGen** ends up with several critical scenarios, e.g. for the values from above:

$$(0,0) \xrightarrow{1} (0,2.5) \xrightarrow{1} (10,2.5) \quad \text{and} \quad (0,0) \xrightarrow{1} (0,-3.5) \xrightarrow{1} (10,-3.5)$$

Two-Dimensional Abstraction. Suppose that the following abstractions are performed: a split of the second segment, and a doubling of the (new) fourth waypoint together with an abstraction of the x-component of the newest waypoint. This yields the abstracted scenario

$$(0,0) \xrightarrow{1} (0,1) \xrightarrow{1} (c,1) \xrightarrow{1} (a,1) \xrightarrow{1} (10,1)$$

where a, c are restricted to $0 \leq c, a \leq 10$. For this abstracted scenario (and maneuver) the Algorithm 3.1 returns within about 70 msec

$$\texttt{Critical} = \texttt{Unsafe}_{(0\times0)} \vee \texttt{Unsafe}_{(1\times0)} \vee \texttt{Unsafe}_{(2\times0)} \vee \texttt{Unsafe}_{(3\times0)}$$

where $\texttt{Unsafe}_{(0\times0)}$ and $\texttt{Unsafe}_{(1\times0)}$ are both `false` and

$$\texttt{Unsafe}_{(2\times0)} = 38 \leq 10c \leq 51 \wedge 0 \leq a \leq 4 \wedge 5a - 10c + 28 \leq 0$$
$$\texttt{Unsafe}_{(3\times0)} = 0 \leq a \leq 4 \wedge -5a + 10c - 31 \leq 0 \wedge 8 \leq 10c - 10a \leq 21$$

These computed constraint formulas are illustrated in Figure 5 where the lambda-shaped red area represents the unsafe scenarios and the top left green triangle represents the original scenario[6]. Note that the red area consists of two parts: The vertical part is responsible for phase (2×0), i. e. situations where the pedestrian is heading towards the other side of the street, but decides fairly late to return. The more diagonal part illustrates the critical a, c-pairs for phase (3×0). Here again, the pedestrian first tries to cross the street, decides pretty early to turn but finally nevertheless returns again for the other side.

[6] Evidently, for any $0 \leq c \leq a \leq 10$ the corresponding instantiation is behavior equivalent to the original scenario. Therefore the safe (green) variants form the top-left triangle instead of a single point.

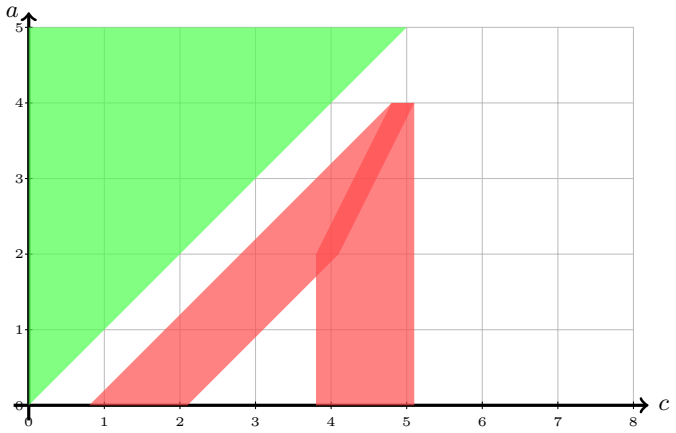

Fig. 5. Unsafe (red λ-shaped) and original (green triangle) scenarios.

There are several interesting points in this region that can serve as candidates for (c, a), like $(5, 3.5), (4, 3), (3, 2), (2, 1), (1, 0)$. For instance, the pairs $(5, 3.5)$ and $(1, 0)$ result in the critical scenarios

$$(0,0) \xrightarrow{1} (0,1) \xrightarrow{1} (5,1) \xrightarrow{1} (3.5,1) \xrightarrow{1} (10,1)$$
$$(0,0) \xrightarrow{1} (0,1) \xrightarrow{1} (1,1) \xrightarrow{1} (0,1) \xrightarrow{1} (10,1)$$

Three-Dimensional Abstraction. Finally, let us perform some further abstraction by replacing the pedestrian's velocity in the final segment with a parameter b. This leads to the abstracted scenario

$$(0,0) \xrightarrow{1} (0,1) \xrightarrow{1} (c,1) \xrightarrow{1} (a,1) \xrightarrow{b} (10,1)$$

After determining the various constraint formulas and after computing all the necessary projections, **CriSGen** produces after about 250 msec the constraint formula for `Critical`, which, in this example, consists of five disjunctions:

$$\begin{aligned}
&\texttt{Critical} = \\
&38 \le 10c \le 51 \wedge 0 \le a \le 4 \wedge 5a - 10c + 28 \le 0 \quad \vee \\
&b > 0 \wedge b < 1 \wedge 5bc - 31b - 5c + 10 \le 0 \wedge 5bc - 28b - 5c + 20 \ge 0 \wedge a = c \quad \vee \\
&a \le 4 \wedge a < c \wedge 5a - 10c + 31 \ge 0 \wedge b > 0 \wedge 5ab + 5a - 10bc + 28b \le 20 \quad \wedge \\
&\qquad\qquad 5ab + 5a - 10bc + 31b - 10 \ge 0 \quad \vee \\
&a \le 4 \wedge a < c \wedge 5a - 10c + 31 \ge 0 \wedge b > 0 \quad \wedge \\
&\qquad\qquad 10 + 10bc - 31b \le 5ab + 5a \le 20 + 10bc - 28b \quad \vee \\
&0 < b < 1 \wedge 5ab - 5a - 31b + 10 \le 0 \wedge 5ab - 5a - 28b + 20 \ge 0 \wedge c \le a
\end{aligned}$$

Since there are three parameters involved, the corresponding unsafe region can be illustrated in a 3D graphic as shown in Figure 6[7]. There are lots of additional

[7] Note that in this illustration the unsafe region and the original region do not intersect, since cutting the area with the plane at $b = 1$ produces exactly Figure 5.

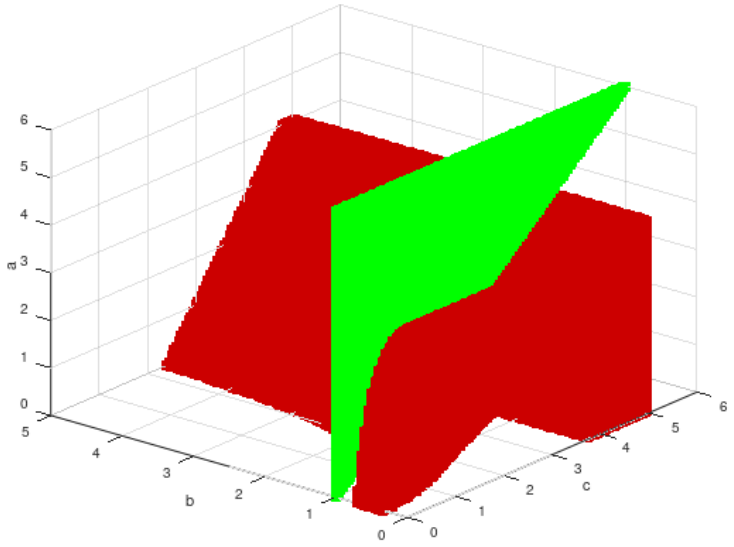

Fig. 6. Unsafe region (red) and original region (green).

interesting points in these unsafe-regions. For instance, $(a, b, c) = (2.5, 0.4, 1)$ and $(a, b, c) = (2.4, 0.5, 2)$, which instantiate the abstracted scenario to the critical scenarios

$$(0,0) \xrightarrow{1} (0,1) \xrightarrow{1} (1,1) \xrightarrow{1} (2.5,1) \xrightarrow{0.4} (10,1)$$

and

$$(0,0) \xrightarrow{1} (0,1) \xrightarrow{1} (2,1) \xrightarrow{1} (2.4,1) \xrightarrow{0.5} (10,1)$$

respectively.

5 Related Work

To our knowledge the **CriSGen** approach is the first that utilizes formal reasoning on non-linear arithmetic constraint formulas with free parameters to synthesize critical scenarios for self-driving cars from previous, similar traffic scenarios and maneuvers. Nevertheless, there exist various alternative, related approaches for scenario generation [14, 19, 2, 13, 8, 9, 11, 20, 10].

In [8] Eggers et al. derive constraint problem classes to be solved for their synthesis, however without showing how to solve them. Their underlying language is based on the graphic representation of Damm et al. [4]. The graphic components get their semantics by a translation into a first-order sorted linear temporal logic which is interpreted in terms of the trajectories of the hybrid automata that represent the vehicles and pedestrians. Althoff & Lutz [1] propose yet another way to automatically generate critical scenarios with the help of formal methods. Whereas we consider a fixed maneuver in a traffic scenario to be challenged, they try to reduce the solution space for maneuvers of the car.

For systematic testing, Frassinelli et al. [9] propose a rule-based mechanism which confronts the autonomous car online while driving with just-in-time generated but not necessarily critical road extensions. Groh et al. [11] discuss transferring the test space into a scenario-depending representation which enables the comparison of scenarios across test domains. Aréchiga [2] use signal temporal logic and Bouton et al. [3] employ reinforcement learning together with a model checker to ensure safety guarantees. In fact, using evolutionary computing or deep learning methods for the generation of critical scenarios is becoming interesting recently such as in Wachi [19, 14]. Other related work focuses on extraction and representation of scenarios. For example, Queiroz et al. [18] propose OpenDSL for scenario representations and Menzel et al. [16] introduce a method to automatically generate executable scenario representations from keyword-based descriptions. Fremont et al. [10] introduce Scenic, a scenario specification language that allows the modeler to mutate scenarios and Yaghoubi & Fainekos [20] determine adversaries for neural network inputs with a gradient descent approach.

6 Conclusion

In this paper, we presented a novel formal method-based approach **CriSGen** for an automated and complete generation of critical traffic scenarios for virtual training of self-driving cars. These scenarios are determined as close variants of given but uncritical and formally abstracted scenarios via reasoning on their non-linear arithmetic constraint formulas, such that the original maneuver of the self-driving car in them will not be pedestrian-safe anymore, hence enforcing it to further adapt the maneuver behavior. The approach is *complete* for the considered scenario abstraction in the sense that, unlike other related methods, it can guarantee to not overlook any of the possible scenario instances that are critical for the original maneuver.

Acknowledgement. This research was supported by the German Federal Ministry for Education and Research (BMB+F) in the project REACT.

References

1. Althoff, M., Lutz, S.: Automatic Generation of Safety-Critical Test Scenarios for Collision Avoidance of Road Vehicles. In: Proceedings of the 29th IEEE Intelligent Vehicles Symposium (IV) (2018)
2. Aréchiga, N.: Specifying Safety of Autonomous Vehicles in Signal Temporal Logic. In: Proceedings of the 30th IEEE Intelligent Vehicles Symposium (IV) (2019)
3. Bouton, M., Nakhaei, A., Fujimura, K., Kochenderfer, M.J.: Safe Reinforcement Learning with Scene Decomposition for Navigating Complex Urban Environments. In: Proceedings of the 30th IEEE Intelligent Vehicles Symposium (IV) (2019)
4. Damm, W., Kemper, S., Möhlmann, E., Peikenkamp, T., Rakow, A.: Traffic sequence charts - from visualization to semantics. Tech. rep., AVACS (2017)

5. DFKI: OpenDS, https://opends.dfki.de/
6. Dolzmann, A., Sturm, T.: redlog, http://www.redlog.eu
7. Dolzmann, A., Sturm, T., Weispfenning, V.: Real Quantifier Elimination in Practice. In: Algorithmic Algebra and Number Theory. Springer (1999)
8. Eggers, A., Stasch, M., Teige, T., Bienmüller, T., Brockmeyer, U.: Constraint systems from traffic scenarios for the validation of autonomous driving. In: Proceedings of Symbolic Computation and Satsfiability Checking (2018)
9. Frassinelli, D., Gambi, A., Nürnberger, S., Park, S.: DRiVERSITY – Synthetic Torture Testing to Find Limits of Autonomous Driving Algorithms. In: Proceedings of the 2nd ACM Computer Science in Cars Symposium (CSCS) (2018)
10. Fremont, D.J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: A language for scenario specification and scene generation. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI). ACM (2019)
11. Groh, K., Kuehbeck, T., Fleischmann, B., Schiementz, M., Chibelushi, C.C.: Towards a Scenario-Based Assessment Method for Highly Automated Driving Functions. In: Proceedings of the 8th Conference on Driver Assistance (2017)
12. Henzinger, T.A., Rusu, V.: Reachability Verification for Hybrid Automata. In: Proceedings of the 1st International Workshop on Hybrid Systems: Computation and Control (HSCC). LNCS 1386, Springer (1989)
13. Jesenski, S., Stellet, J.E., Schiegg, F., Zöllner, J.M.: Generation of Scenes in Intersections for the Validation of Highly Automated Driving Functions. In: Proceedings of the 30th IEEE Intelligent Vehicles Symposium (IV) (2019)
14. Klischat, M., Althoff, M.: Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms. In: Proceedings of the 30th IEEE Intelligent Vehicles Symposium (IV) (2019)
15. Lamport, L.: Hybrid systems in TLA+. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) Hybrid Systems. Springer (1993)
16. Menzel, T., Bagschik, G., Isensee, L., Schomburg, A., Maurer, M.: From Functional to Logical Scenarios: Detailing a Keyword-Based Scenario Description for Execution in a Simulation Environment. In: Proceedings of the 30th IEEE Intelligent Vehicles Symposium (IV) (2019)
17. Pusse, F., Klusch, M.: Hybrid Online POMDP Planning and Deep Reinforcement Learning for Safer Self-Driving Cars. In: Proceedings of the 30th IEEE Intelligent Vehicles Symposium (IV). IEEE (2019)
18. Queiroz, R., Berger, T., Czarnecki, K.: GeoScenario: An Open DSL for Autonomous Driving Scenario Representation. In: Proceedings of the 30th IEEE Intelligent Vehicles Symposium (IV) (2019)
19. Wachi, A.: Failure-scenario maker for rule-based agent using multi-agent adversarial reinforcement learning and its application to autonomous driving. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI) (2019)
20. Yaghoubi, S., Fainekos, G.: Gray-box adversarial testing for control systems with machine learning components. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC) (2019)