Preventing Collisions in Self-driving Cars using Probabilistic Logic Counterfactual Reasoning

Verónica Rodríguez[1 [0000−0002−5976−9338]], Héctor Avilés[2 [0000−0001−5310−3474]], Rubén Machucho[2 [0000−0002−5731−6677]], Alberto Reyes[3 [0000−0002−8509−6974]], Marco Negrete[4 [0000−0002−5468−2807]], Gabriel Ramírez[5 [0000−0001−5226−5615]], Alberto Petrilli[6 [0000−0002−2766−7599]], Ingridh Gracia[2 [0009−0004−7101−6656]], Gloria De-La-Garza[2 [0009−0004−2446−6435]], Karelly Rivera[2 [0000−0002−4749−0663]], and Rafael Kiesel[7 [0000−0002−8866−3452]]

[1] Technological University of the Mixteca, 69004 Oaxaca, Mexico
veromix@mixteco.utm.mx
[2] Polytechnic University of Victoria, 87138 Tamaulipas, Mexico
{havilesa,rmachuchoc,2330319,2130071,1930435}@upv.edu.mx
[3] National Institute of Electricity and Clean Energy, 62490 Cuernavaca, Mexico
areyes@ineel.mx
[4] National Autonomous University of Mexico, 04510 Mexico City, Mexico
marco.negrete@ingenieria.unam.edu
[5] Center for Research and Advanced Studies of the National Polytechnic Institute, Tamaulipas Campus, 87138 Tamaulipas, Mexico
grtorres@cinvestav.mx
[6] Tohoku University, 980-8577 Miyagi, Japan
a.petrilli@srd.mech.tohoku.ac.jp
[7] Vienna University of Technology, 1040 Vienna, Austria
rafael.kiesel@tuwien.ac.at

Abstract. We propose counterfactual reasoning through probabilistic logic twin networks (*PLTNs*) to prevent collisions in self-driving cars. The basis of a PLTNs is a causal Bayesian network (*cBN*) partially learned from simulated self-driving car data and synthetic data. The cBN include the lane of the self-driving car, the presence of up to 4 surrounding vehicles, an indicator for potential collisions, and 6 driving actions. Counterfactual queries through the PLTNs intervene with alternative actions to identify which minimizes the probability of a collision. For evaluation, three cBNs are learned with 1%, 50%, and 100% of a training dataset. For querying, 120 state-action examples labeled as leading to a crash are selected randomly. Each one is associated with six possible interventions. The probability of a collision is then queried from PLTNs, provided that a potential collision has been warned, and the current state and action are known. Results show that all intervened actions minimizing the probability of a crash does not lead to a car crash, suggesting the effectiveness of this approach in developing collision prevention schemes for self-driving cars. To the best of our knowledge, this is the first application of PLTNs for counterfactual reasoning in autonomous vehicles.

Keywords: Counterfactual reasoning · probabilistic logic twin networks · Autonomous vehicles.

1 Introduction

Collision prevention is an important concern in self-driving cars. Although diverse approaches exist to achieve this goal, many rely on adjusting the current trajectory or behavior of the self-driving car to a new, safer course of action [19–21]. In this context, we propose probabilistic counterfactual queries to pose questions about prospective, hypothetical situations like:

"*What is the probability that a situation of potential collision could prevail, given that the current action and the state of the system are known, if a different action were chosen?*"

Answers to that kind of "*What-if*" questions can help self-driving cars evaluate the convenience of alternative driving actions to perform safer maneuvers under hazardous situations [16]. Additionally, the probabilistic causal models associated with causal questions can be framed in probabilistic logic programming [15], a paradigm well-suited for modeling causal relationships due to both its clear and flexible rule-based representation, and the availability of sophisticated inference procedures to efficiently solve probabilistic queries on complex probability distributions [6, 8].

Therefore, in this paper, we present our ongoing work on counterfactual reasoning using probabilistic logic twin networks (*PLTNs*) as a counterfactual model to identify alternative actions that may prevent collisions in self-driving cars. PLNTs are twin networks [2] encoded as probabilistic logic programs. Inference is carried out through Counterfactuals [11], an efficient and effective tool recently developed to solve probabilistic counterfactual queries in PLTNs. Causal Bayesian networks (*cBN*) are proposed as the basis of the PLTNs. Three cBNs are constructed, for the most part, from a large dataset of examples taken from multiple runs of simulations of our self-driving car under autonomous and human control, and from synthetic data. The state variables included in the cBNs are the lane in which the self-driving car is traveling and six "occupancy" variables that indicate the presence of surrounding vehicles. On each state, the self-driving car can choose one of six possible actions. When a potential collision is warned, the goal is to calculate the probability of the collision given the current state and action, by intervening in the causal model with alternative actions, one at a time. It is assumed that the counterfactual action minimizing the probability of the collision is the best alternative for the self-driving car to avoid or mitigate the severity of the impact.

To evaluate our approach, three cBNs were constructed using 1%, 50%, and 100% of a training dataset containing over 1,900,000 examples of state-action pairs. For testing, 120 examples were randomly selected from a pool of 288 state-action pairs labeled as potential crashes. Each of these examples was subsequently associated with six alternative actions implemented in our simulated self-driving car, forming a group of six examples that encompass the state, the observed action, and possible interventions. Examples within each group are independently queried to a PLTN to compute its corresponding probability value of a crash.

A qualitative analysis of the results shows that, in all 120 cases the counterfactual actions with the lowest probability of potential collision within each group does not lead to a crash in any of the PLTNs. This finding suggests the viability of the proposed approach for developing a collision avoidance module in self-driving cars.

2 Related work

Recent research into autonomous vehicles (*AVs*) and their safety has made significant strides using counterfactual reasoning, what-if analysis, and advanced simulation techniques to address collision risks and enhance traffic safety. Studies have employed counterfactual simulations to evaluate the performance of AVs in preventing collisions and improving safety. For example, research on the Waymo Driver [17] demonstrated its effectiveness in avoiding fatal collisions by simulating crash scenarios and showing that it could prevent or mitigate a substantial percentage of crashes. Similarly, the impact of Advanced Driver Assistance Systems (*ADAS*) has been analyzed using counterfactual reasoning to assess their safety benefits and drawbacks in real-world scenarios [3]. This approach allows for a nuanced understanding of how different safety technologies might alter crash outcomes and driver behavior. Another relevant study focused on predicting crash configurations and the impact of Autonomous Emergency Braking (*AEB*) systems [12]. This research utilized counterfactual simulations to identify specific crash scenarios that AEB systems could not address, thus highlighting areas for improvement. Additionally, counterfactual reasoning has been applied to estimate the importance of objects in autonomous driving environments [10], enhancing the system's ability to prioritize critical objects and reduce collision risks. The effectiveness of pre-crash safety technologies, such as AEB and Electronic Stability Control (*ESC*), has also been evaluated in the context of reducing severe injuries in crashes [14]. This study underscores the role of these technologies in mitigating collisions and emphasizes the need for continued advancements to address remaining safety concerns. Furthermore, research on the design and evaluation of Automated Driving Systems (*ADS*) has utilized injury risk modeling to understand the potential outcomes of various crash scenarios [13]. By developing comprehensive injury risk surfaces, this research provides valuable insights into how different ADS designs could impact overall safety. The importance of considering driver behavior models in counterfactual simulations has been highlighted as well in [5]. Different models can significantly influence the effectiveness estimates of safety systems, underscoring the need for accurate simulations to evaluate safety technologies properly.

3 Methodology

3.1 Testbed and datasets

Our development framework involves a self-driving car simulated in race-like environments using the Webots simulator. An example of the self-driving car and

Fig. 1. Race-like environment considered in this study for our self-driving car (in bright red).

its environment is shown in Fig. 1. The car can travel on a two-lane road with straight segments and curves and up to 10 obstacle vehicles distributed over the road and either static or in motion. However, in this work, we are considering straight roads only. It is assumed that the vehicles traveling on the left lane moves faster than those on the right lane, while the self-driving car is the fastest (the maximum tested speed is near 50 km/h).

The architecture of the self-driving vehicle includes modules for *perception*, *driving control* and *decision making*. Perception utilizes an accelerometer for collision detection, 3D laser readings for detecting other vehicles, and a RGB camera to detect lane borders. The control module is responsible for lane tracking, and speed and steering estimation, following standard control laws. Decision making selects driving behaviors or actions based on a state-action policies estimated by solving probabilistic logic factored Markov decision processes (*PL-MDPs*) [1, 4]. These policies primarily promote traveling on the right lane while using the left lane only for overtaking. For decision making, the vehicle behind the self-driving car in the same lane is not explicitly considered. State variables are `curr_lane` which identifies the current lane of the self-driving car, and occupancy variables called `free_E`, `free_NE`, `free_NW`, `free_SE`, `free_SW` and `free_W` which indicate whether there is a vehicle or not in the location indicated in the name of each variable, relative to the self-driving car. Figure 2 depicts the locations represented by the occupancy variables on each lane. In this work, we incorporate a new variable called `latent_collision` used to warn a potential collision in the trajectory of the self-driving car. All these variables are binary. Finally, a multi-valued variable called `action` identifies one of four driving maneuvers for the self-driving car: **change_to_left** and **change_to_right** used for overtaking on the left and returning to the right lane, respectively, **cruise** to reach a steady (maximum) speed, **keep** to maintain a safe, steady distance to a vehicle ahead in the same lane, and **swerve_right** and **swerve_left** which perform a controlled veer to the side of the lane while reducing speed. We consider the two latter actions also as safe, evasive maneuvers in case of unexpected situations.

We have recorded several runs of the self-driving car system under both autonomous control and human command via a human-friendly interface developed on purpose[8]. These datasets include 1,238,869 driving decisions through

[8] These datasets are available at: https://www.kaggle.com/autonomousvehicle/

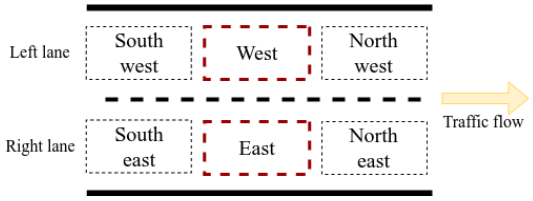

Fig. 2. Predefined locations for other vehicles around the self-driving car (dashed red lines indicate the space the self-driving car occupy on each lane). When the vehicle is on the left (resp. right) lane, only the locations Northwest, Northeast, East and Southeast (resp. Northeast, Northwest, West and Southwest) are meaningful.

more that 300 km. The source code of our self-driving vehicle and some videos of the self-driving system running are available at: https://github.com/hector-aviles/AIRJ2024/.

The approach used in this work requires labeling each state-action pair in the dataset as either leading to a collision or not. In the proposed driving environment, we consider that a crash may occur principally in two situations: (a) when the self-driving car applies the action **cruise** and there is a vehicle ahead in the same lane (rear-end collision), or (b) when there is a car either next to or ahead in the lane the self-driving car merges into (sideswipe and rear-end crash, respectively). Following the previous criteria, from the complete space of $2^7 \times 6 = 768$ state-action pairs, we carefully labeled by hand 288 pairs as leading to a potential collision. Some tate-action pairs that involve **change_to_left**, **change_to_right** and **cruise** emerged as the "unsafe" driving combinations. With this new list as reference, we found that the automated and human control databases contain only a small number of potential crash examples (39,695 pairs, all of which are repetitions of 132 unique pairs). Unfortunately, as of the time of writing, sampling potential crashes by human control has not been completed due to its tedious and time-consuming nature. Instead, the 288 state-action pairs were replicated 2,500 times to generate a synthetic dataset of 720,000 potential crashes. The number of repetitions was selected arbitrarily, with the aim of approximating a balance between collision and non-collision examples. During the process, continuous variables such as the speed and steering of the self-driving car were random sampled from their known data distributions, although they are not considered in the present work. This new dataset of crashes complements the driving records originally obtained from the autonomous and human control of the self-driving car. Each example in the integrated dataset was labeled using `latent_collision` as a potential collision or not. Table 1 summarizes the number of actions labeled as potential collisions or non-potential collisions in the complemented dataset. Table 2 shows the number of unique state-action pairs on each subset.

Table 1. Number of actions in state-action pairs labeled as potential collisions and non-potential collisions in the integrated dataset.

Actions	# of examples of non-potential collisions	# of examples of potential collisions	**Total:**
change_to_left	40,308	289,809	330,117
change_to_right	39,814	294,459	334,273
cruise	617,930	175,429	793,359
keep	496,558	0	496,558
swerve_left	2,883	0	2,883
swerve_right	1,681	0	1,681
Total:	1,199,174	759,695	1,958,869

Table 2. Number of unique state-action pairs labeled as non-potential collisions and potential collisions ("safe" and "unsafe", respectively).

Actions	# of unique "safe" state-action pairs	# of unique "unsafe" state-action pairs	**Total:**
change_to_left	14	112	126
change_to_right	15	112	127
cruise	53	64	117
keep	67	0	67
swerve_left	36	0	36
swerve_right	29	0	29
Total:	214	288	502

3.2 Learning of the causal Bayesian networks

Structural and parameter learning of cBNs were instrumented with the **bnlearn** package in R package [18]. Its directed acyclic graph is shown in Fig. 3. Structural learning was carried out by means of hill-climbing greedy search with Bayesian information criterion. Initially, we devised a preliminary graph that we wished to test, so undesired links between variables were forbidden (for instance, those between occupancy variables, or from `action` to `curr_lane`). In contrast, only the relation that goes from `action` to `latent_collision` was explicitly requested to be included, while letting hill-climbing to decide about the rest of the connections. We believe the resulting structure resembles a causal graph with confounders, with `action` being the treatment variable, `latent_collision` being the output variable and the current lane and occupancy variables jointly playing the role of a single, multi-valued confounder. The parameters of the cBN were fitted by the *maximum likelihood estimation* criterion.

For training, we considered the integrated dataset containing 1,958,869 examples. Training data is organized into `state-action-latent_collision` triplets. From our perspective, data leakage (that may occur when non-disjoint datasets

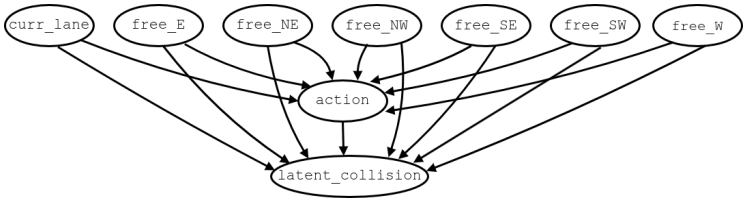

Fig. 3. Causal Bayesian network (cBN) proposed in this work.

Table 3. Number of unique state-action pairs as a function of the action used for training and testing each cBN.

Action	Training dataset			Test dataset
	1%	50%	100%	
change_to_left	122	126	126	49
change_to_right	118	126	127	46
cruise	101	114	117	25
keep	31	62	67	0
swerve_left	7	32	36	0
swerve_right	4	29	29	0

are used for training and testing) does not represent a major concern in this setup. This is because the core of the evaluation focuses on comparing conditional probabilities under interventions for a counterfactual model, rather than studying the generalization capabilities of PLTNs to previously unknown data. Despite this, we consider constructing cBNs using training sets of varying sizes for comparison purposes. To achieve this, we randomly selected 1%, 50%, and 100% of the training dataset for structural and parameter learning of the cBNs. Table 3 summarizes the number of unique state-action pairs used for training and testing for each driving action and cBN.

3.3 Counterfactual queries

Counterfactual querying requires a cBN to be provided as a parameter to the Counterfactuals library, formatted as a probabilistic logic program in ProbLog syntax [9]. To achieve this, we developed a purpose-built R script to syntactically translate the cBN learned with **bnlearn** into a ProbLog program. Although this conversion results in a large format size in comparison to the well-known compactness of traditional conditional probability tables, we have found that the probabilistic logic representation of a cBN as a set of probabilistic facts and rules was convenient during development for inspection, debugging and communication. In the Counterfactuals library, two copies of the ProbLog program are required to represent both the real and the counterfactual worlds. The two ProbLog programs have the same rules as the original program and share probabilistic facts representing not observable external factors. To obtain the twin

networks, each rule of the original ProbLog program is duplicated, labeling the variables on each copy with the superscript **e** for the real, and **i** for the counterfactual world, respectively. As an example, consider the following Counterfactuals representation of a twin network inspired in the model proposed in this work:

```
% Probabilistic facts
0.5349662::u1.
0.7046031::u2.
0.4785868::u3.
0.1143778::u4.

% Rules for the real world
free_NE^e :- u1.
free_W^e :- u2.
action(keep)^e :- free_NE^e, free_W^e, u3.
latent_collision^e :- action(keep)^e, free_NE^e, free_W^e, u4.

% Rules for the counterfactual world
free_NE^i :- u1.
free_W^i :- u2.
action(keep)^i :- free_NE^i, free_W^i,u3.
latent_collision^i :- action(keep)^i, free_NE^i, free_W^i, u4.
```

A twin ProbLog program of this type constitutes a counterfactual model, which is used to solve probabilistic counterfactual queries. A counterfactual query is evaluated by first absorbing the evidence in the real ProbLog rules, then performing the intervention in the counterfactual ProbLog rules, and finally, using the internal inference engine of Counterfactuals [7] to assess the conditional probability of a hypothetical event given the observed values of certain variables as evidence. This scheme allows us to perform queries about the probability value of any variable of the cBN being true, given the evidence available, along with interventions on the truth values of one or more variables. In our case, the query is focused on the variable `latent_collision`. The evidence includes the current lane, the occupancy variables and the action it is performed. The counterfactual variable is `action`. Thus, we are interested in calculating probability values of the type[9]:

$$\begin{aligned} P\big(&\texttt{latent\_collision}^i = \mathbf{T} \big| \texttt{latent\_collision}^e = \mathbf{T}, \texttt{curr\_lane}^e = \mathbf{right}, \\ &\texttt{free\_E}^e = \mathbf{T}, \texttt{free\_NE}^e = \mathbf{F}, \texttt{free\_NW}^e = \mathbf{F}, \texttt{free\_SE}^e = \mathbf{T}, \\ &\texttt{free\_SW}^e = \mathbf{T}, \texttt{free\_W}^e = \mathbf{F}, \texttt{action}^e = \mathbf{cruise}, \\ &do(\texttt{action}^i = \mathbf{keep})\big) \end{aligned} \tag{1}$$

[9] The truth value **T** means the truth-value true, and **F** means false (in the case of the occupancy variables, **T** indicates that a location is empty and **F** that it is occupied by another vehicle).

Table 4. Number of groups with a unique minimum probability value and the number of ties for first place (ranging from 2 to 5) across all groups and counterfactual models.

# of ties (1^{st} place)	# of cases (1%)	# of cases (50%)	# of cases (100%)	**Total:**
No ties	43	23	20	86
2	32	32	33	97
3	28	38	42	108
4	15	26	24	65
5	2	1	1	4
Total:	120	120	120	360

4 Evaluation and results

The evaluation of a counterfactual model involves a random selection of 120 unique `state-action-latent_collision` triplets from the list of identified potential collisions (that is, in which `latent_collision` = **T**). These triplets are further expanded by adding an intervention over the observed action with each of the six actions that the self-driving car can perform (including the action originally observed). Each unique triplet is thus transformed into a group of six `state-action-latent_collision-intervention` quartets, all sharing the same observed `state-action-latent_collision` triplet, but incorporating a different intervention on the action. From each quartet, a query similar to that in Equation 1 is derived to be solved by the counterfactual models. In total, each counterfactual model solves 720 queries. The objective is to identify an action (among the six intervention alternatives) that minimizes the probability of `latent_collision`. It is assumed that the counterfactual actions with the lowest probability of `latent_collision` represents the best decision at hand to prevent or reduce the severity of an accident under the observed circumstances.

Table 4 presents the number of ties for the first place observed across the 120 groups for the three counterfactual models when ranking the quartets within each group from lowest to highest, according to their probability values. We consider there can be from 2 to 5 ties on each group (the action currently observed has a probability value of 1 and there is at least one safe action on each state). For tie-breaking (that is, selecting an appropriate intervention among those sharing the lowest probability value), we implemented a straightforward, standard approach based on random selection. In accordance with our initial labeling of state-action pairs as potential or non-potential collisions, results show that in all three counterfactual models and test examples, the intervened action associated to the lowest probability value for `latent_collision` may indeed prevent a crash. Table 5 resume the number of actions selected as the best alternative for each counterfactual model, following the previous scheme. The average time for solving each counterfactual query on the three models is 5.74 seconds (SD=0.15). All tests were performed on a standard Core i7 laptop computer.

Table 5. Number of actions selected as the optimal intervention in the three counterfactual models.

Action	# of best interventions (1%)	# of best interventions (50%)	# of best interventions (100%)	**Total:**
change_to_left	1	1	0	2
change_to_right	0	4	2	6
cruise	23	25	22	70
keep	59	51	47	157
swerve_left	22	24	30	76
swerve_right	15	15	19	49
Total:	120	120	120	360

Table 6. State-action pairs with 5 tied actions. The first two pairs, from top to bottom, correspond to the counterfactual model trained with 1% of the data, and the last one is the same for the models trained with 50% and 100% of the data (all actions other than the observed are safe).

State variables							Observed action
`curr_lane`	`free_E`	`free_NE`	`free_NW`	`free_SE`	`free_SW`	`free_W`	
right	**F**	**T**	**T**	**F**	**T**	**T**	*CTR*
right	**F**	**T**	**T**	**F**	**F**	**T**	*CTR*
left	**T**	**T**	**T**	**F**	**F**	**T**	*CTL*

4.1 Discussion

The initial results described above are encouraging. A qualitative analysis revealed that 100% of the best driving decisions obtained through counterfactual reasoning prevent car collisions. When ties occur between alternative interventions, choosing an appropriate action for the current driving scenario becomes particularly important. In our case, this is especially relevant due to the significant number of ties identified on closer inspection of the results. A thorough analysis of the ranking of `state-action-latent_collision-intervention` quartet with respect to their probability values within each group demonstrated that, in all groups, all interventions with the lowest probability value are also safe actions (note that there may be more than one safe action for some states). For example, Table 6 show the 3 unique cases in which there are 5 equivalent interventions. Yet factors beyond safety that include efficiency on energy consumption, better use of available space, better adherence to traffic rules, cooperation with other vehicles and human preferences can also be taken into account to improve tie-breaking.

5 Conclusions and future work

We have presented a counterfactual reasoning approach to prevent collisions in a self-driving car using probabilistic logic twin networks. Twin networks are derived from causal Bayesian networks designed through expert knowledge, data from simulations and synthetic data. Three causal Bayesian networks were constructed and tested using training sets of varying sizes. Counterfactual reasoning allowed us to decrease the probability of a car crash by intervening with alternative maneuvers, yielding promising results. Although we have not yet fully exploited the descriptive and inferential capabilities of the Counterfactuals library, we recognize its suitability for development, debugging and analysis. This probabilistic logic approach does not only enhance clarity of the causal model, but it help us consider its deductive capabilities in explainability tasks in which it is useful to track the inference path.

As a future work, we plan to perform an exhaustive testing with more collision cases, make more extensive use of the pobabilistic logic approach to include new information for tie-breaking, and integrate counterfactuals into the decision-making module of our self-driving vehicle. Additionally, we will incorporate new information to the causal model, such as discrete versions of the distance from the self-driving car to other vehicles, as well as speed and steering of all the vehicles on the road. Furthermore, we will explore Counterfactuals to achieve counterfactual explanations about relevant variables and rules in decision-making, with the primary aim of enhancing transparency and trustworthiness in autonomous vehicle systems.

Acknowledgments. This work was partially founded by UNAM-DGAPA under grant IT102424 and AI Consortium - CIMAT-CONAHCYT.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Avilés, H., Negrete, M., Reyes, A., Machucho, R., Rivera, K., de-la Garza, G., Petrilli, A.: Autonomous Behavior Selection For Self-driving Cars Using Probabilistic Logic Factored Markov Decision Processes. Applied Artificial Intelligence **38**(1), 2304942 (2024)
2. Balke, A., Pearl, J.: Probabilistic evaluation of counterfactual queries, p. 237–254. Association for Computing Machinery, New York, NY, USA, 1 edn. (2022)
3. Bärgman, J., Victor, T.: Holistic assessment of driver assistance systems: how can systems be assessed with respect to how they impact glance behaviour and collision avoidance? IET Intelligent Transport Systems **14**(9), 1058–1067 (Aug 2019)
4. Bueno, T.P., Mauá, D.D., De Barros, L.N., Cozman, F.G.: Markov decision processes specified by probabilistic logic programming: representation and solution. In: 2016 5th Brazilian Conference on Intelligent Systems (BRACIS). pp. 337–342. IEEE (2016)

5. Bärgman, J., Boda, C.N., Dozza, M.: Counterfactual simulations applied to shrp2 crashes: The effect of driver behavior models on safety benefit estimations of intelligent safety systems. Accident Analysis & Prevention **102**, 165–180 (2017)
6. Chavira, M., Darwiche, A.: On probabilistic inference by weighted model counting. Artificial Intelligence **172**(6-7), 772–799 (2008)
7. Eiter, T., Hecher, M., Kiesel, R.: aspmc: An algebraic answer set counter. In: ICLP Workshops (2021), https://ceur-ws.org/Vol-2970/plppaper1.pdf
8. Eiter, T., Hecher, M., Kiesel, R.: aspmc: New frontiers of algebraic answer set counting. Artificial Intelligence **330**, 104109 (2024)
9. Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., Janssens, G., De Raedt, L.: Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas. Theory and Practice of Logic Programming **15**(3), 358–401 (2015)
10. Gupta, P., Biswas, A., Admoni, H., Held, D.: Object Importance Estimation using Counterfactual Reasoning for Intelligent Driving. IEEE Robotics and Automation Letters **9**(4), 3648–3655 (2024)
11. Kiesel, R., Rückschlo, K., Weitkämper, F.: "What if?" in Probabilistic Logic Programming. Theory and Practice of Logic Programming **23**(4), 884–899 (2023)
12. Leledakis, A., Lindman, M., Östh, J., Wågström, L., Davidsson, J., Jakobsson, L.: A method for predicting crash configurations using counterfactual simulations and real-world data. Accident Analysis & Prevention **150**, 105932 (2021)
13. McMurry, T.L., Cormier, J.M., Daniel, T., Scanlon, J.M., Crandall, J.R.: An omnidirectional model of injury risk in planar crashes with application for autonomous vehicles. Traffic Injury Prevention **22**(sup1), S122–S127 (2021), pMID: 34402345
14. Puthan, P., Lubbe, N., Davidsson, J.: Characterizing future crashes on indian roads using counterfactual simulations of pre-crash vehicle safety technologies. IATSS Research **46**(4), 479–491 (2022)
15. Riguzzi, F.: Foundations of probabilistic logic programming. River Publishers, New York (May 2023)
16. Ruiz-Tagle, A., Lopez-Droguett, E., Groth, K.M.: A novel probabilistic approach to counterfactual reasoning in system safety. Reliability Engineering & System Safety **228**, 108785 (2022)
17. Scanlon, J.M., Kusano, K.D., Daniel, T., Alderson, C., Ogle, A., Victor, T.: Waymo simulated driving behavior in reconstructed fatal crashes within an autonomous vehicle operating domain. Accident Analysis & Prevention **163**, 106454 (2021)
18. Scutari, M.: Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software **35**(3), 1–22 (2010)
19. Sharma, P., Rana, C.: Artificial intelligence based object detection and traffic prediction by autonomous vehicles – a review. Expert Systems with Applications p. 124664 (2024)
20. Tolba, M.A., Kamal, H.A.: SDC-Net++: End-to-End Crash Detection and Action Control for Self-Driving Car Deep-IoT-Based System. Sensors **24**(12), 3805 (2024)
21. Zhang, N., Tian, R., Fu, G.: Design and application of automatic driving emergency collision avoidance control algorithm based on artificial intelligence technology. Measurement: Sensors p. 101248 (2024)