Domain-specific probabilistic programming with Multiverse Explorer

Alan F. Blackwell
Computer Laboratory
University of Cambridge
Cambridge, UK
afb21@cam.ac.uk

Alex Raymond
Computer Laboratory
University of Cambridge
Cambridge, UK
raymond.alex@gmail.com

Colton Botta
Engineering Department
University of Cambridge
Cambridge, UK
cgb45@cam.ac.uk

Matthew Keenan
Computer Laboratory
University of Cambridge
Cambridge, UK
mck36@cam.ac.uk

William Hayter-Dalgliesh
Computer Laboratory
University of Cambridge
Cambridge, UK
willhd2000@gmail.com

***Abstract*—We present Multiverse Explorer, a domain-specific probabilistic programming language presented as a visual language integrated with a domain world model. The interactive visualisation presents a Monte Carlo simulation over a causal graph, allowing the user to gain an overview and query alternative outcomes in a counterfactual manner. Separate graphs express the policies attributed to multiple heterogeneous agents. The outcomes of actions are visualised in an interactive 3D animation of the environment; in this work, we apply the Multiverse Explorer to multi-agent driving scenarios by extending the CARLA simulator. The Multiverse Explorer has been evaluated with a sample of technical non-specialists, demonstrating the potential of this approach to be used in design, audit, policy, litigation, and other contexts where the outcome of multi-agent decision scenarios must be investigated by professionals beyond a specialist AI audience.**

***Index Terms*—Programming, Visualisation, User centered design, Graphical user interfaces**

I. Introduction

Probabilistic programming languages (PPLs) implement a programming paradigm in which the programmer specifies a probabilistic model as a set of relationships between random variables, and execution involves running an inference algorithm to characterise and sample from the resulting distributions. Since the original development of the BUGS language [27] as an implementation of Judea Pearl's Bayesian causal graphs [22], there has been an explosion in development and deployment of PPLs, especially for use as statistical tools [9], [21], [27], and in AI applications that combine symbolic reasoning and machine-learning functionality [12], [16], [30] (sometimes described as "neuro-symbolic") [13]. In terms of historical programming paradigms, probabilistic programming is perhaps most closely related to logic programming, which also involves declarative specification of a model as relations between variables, and execution by running an inference algorithm to establish possible values for variables that are unknown.

As yet, there has been very little research into the human-centric design of PPLs. An agenda-setting paper was presented at the 2019 workshop of the Psychology of Programming Interest Group (PPIG), with authors including several developers of leading PPL projects [2]. The project that we report here builds on the recommendations of that paper. In particular, we explore the potential of the PPL approach to be offered to end-user developers (EUD). Following a common research strategy in EUD, we have created an experimental PPL that is targeted at a specific domain, illustrating how a programming language approach can deliver enhanced capabilities to users working in that domain. We suggest that this class of tools should be described as domain-specific PPLs, or DS-PPLs. We believe that DS-PPLs can become a valuable addition to the set of machine learning tools available to end-users, offering enhanced controllability, explainability, and learnability for machine learning applications.

The DS-PPL that we present in this paper offers the following contributions: 1) a novel visual syntax based on a temporally-ordered causal graph, where line widths indicate likelihood; 2) a novel visual rendering of the world model, where multiple possible outcomes are shown as semi-transparent "ghosts" of moving agents; 3) a novel interactive environment where alternative sequences of events specified by the causal graph are animated as ghosts in the rendered world model; and 4) a novel interaction paradigm for counterfactual model exploration and explanation, where the causes and likelihoods of possible outcomes can be explored using queries within this environment.

We call the combination of these innovations the *Multiverse Explorer* paradigm. This paradigm could be applied to many different DS-PPLs, but we illustrate them in a single domain case study: understanding interaction between multiple

This research was funded by Boeing Research and Technology, project grant *Video Analytics (Autonomous Coordination Task)*. Pilot versions of the user study, not otherwise reported, were conducted by Satyajit Das, Mark Quinlan, and Benjamin Mbuu Mutua

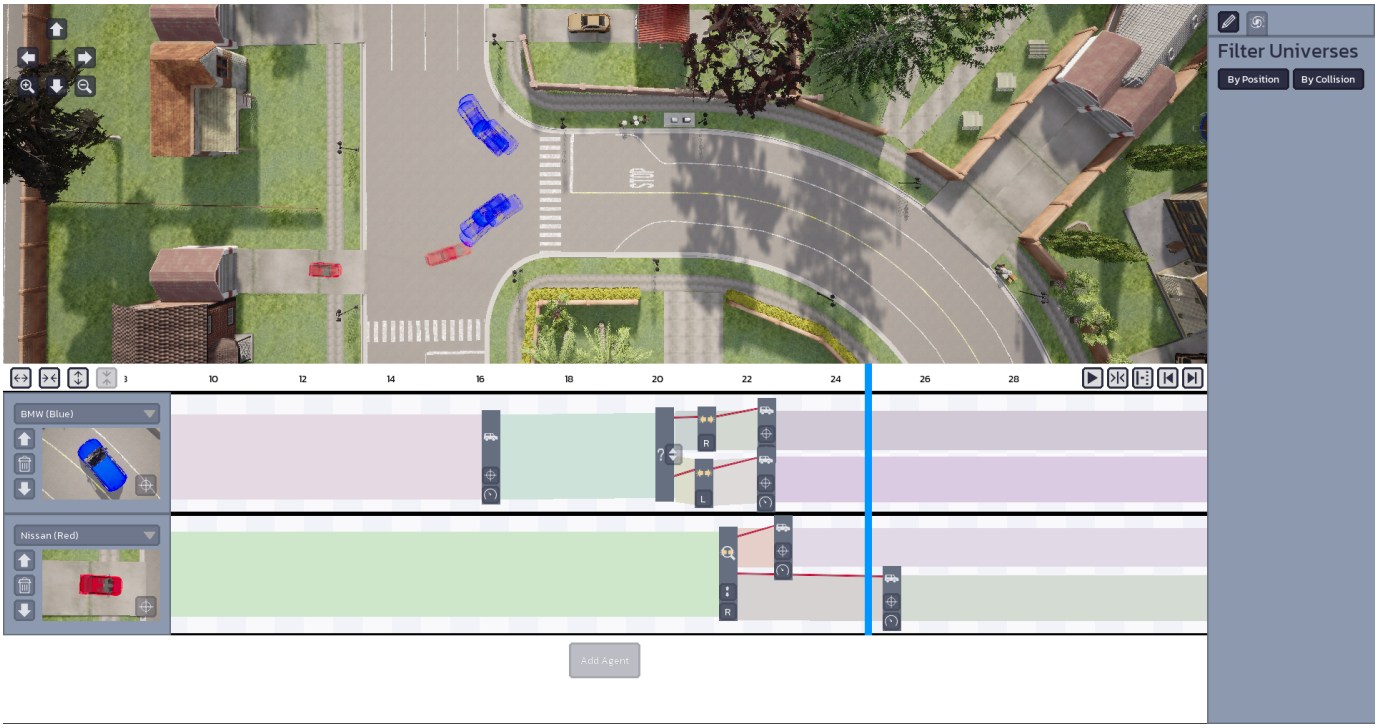


Fig. 1. Overview of the Multiverse Explorer prototype, showing the main elements of the user interface in a simple scenario involving two vehicles.

autonomous vehicles in an urban environment, where the vehicles are guided by different decision policies. We conducted a user study where participants (n=18) used Multiverse Explorer to explain behaviors and outcomes in this domain. We found evidence that our approach assists users in reasoning about the domain, and suggest design principles and opportunities for future DS-PPLs.

We discuss related work, including design precedents that our novel contributions have built on, toward the end of the paper.

II. Overview of Functionality and Features

The Multiverse Explorer[1] is an interactive application consisting of two linked visualisations (Fig 1) which we refer to as the *rendering* and the *timeline*.

A. Rendering

The rendering, which denotes the upper half of the display, is a 3D representation of the world in which the agents are acting. In our current implementation, the rendering shows city streets with vehicles driving along them. Generalisation of the Multiverse Explorer paradigm in the future could visualise any situation with interacting agents, whether realistic rendering as in our case study, or symbolic rendering of more abstract domains. Our choice of a driving simulator, where multiple agents are controlling vehicles, was determined in part by the availability of the open source driving simulator CARLA [14], which we have extended with the multiverse rendering functionality described below. Although the domain itself is not the main focus of our own research, it is well understood that reasoning about the behaviour of other agents is a fundamental problem in driving [6], [7], and that there is a significant challenge in explaining the behaviour of autonomous vehicles to people, including their own drivers [10].

B. Timeline

The timeline, which denotes the lower half of the display, visualises agent decisions in an augmented causal graph. Each horizontal band in this visualisation defines the decisions of a single agent, which can be compared to the "swimlane" convention for visualising events in parallel processes (e.g. the UML sequence diagram, or GANTT charts). The left-to-right dimension corresponds to time and the blue, vertical cursor indicates the point in time at which events in the rendering are paused. Time is mapped as a metrical axis, meaning that locations along the swimlane are not only *ordered* in time (to reflect causality), but *relative* durations separating them can be judged by comparing distances between points on the display. The rectangular nodes in each swimlane represent points at which the agent has either observed some occurrence in the world, or formed an intention to act. As for any other causal graph, connections between the nodes can be considered to represent information flow within a Bayesian process model [28] - for example, an agent may form an intention such as slowing down the car, on the basis of something it has observed, such as another car turning in front of it as seen in Fig. 1. Generalisation of the Multiverse Explorer paradigm in

[1]A video demo is available at
https://www.cl.cam.ac.uk/~afb21/publications/multiverse-explorer/

the future could use these temporally-ordered causal graphs to model many other types of probabilistic behaviour that are conditioned on likelihood of observations.

C. Multiverse Representation

The functionality described so far can be used to visualise a single course of events, such as a car driving and making a turn. However, the core functionality of the Multiverse Explorer is to visualise *multiple* possible courses of events, rather than a single set of events happening concurrently. The nodes do not represent deterministic specifications of behaviour that have been defined to take place at that time, but potential events, where outcomes are conditioned on prior (uncertain) observations and physical processes, as well as the policies or biases that make any given agent likely (with some uncertainty) to choose particular actions and responses following a given observation.

D. Underlying Model

The computational model of the Multiverse Explorer is a Monte Carlo simulation, in which a possible series of events is determined by sampling from each of the probability distributions associated with the random variables in the timeline [19]. In the current configuration, the simulation is run 1000 times, resulting in 1000 possible histories of interaction between the multiple agents over the timeline. Where there are branches in the causal graph that are dependent on the values of particular observations, the number of history paths in each branch is counted, and the width of each line adjusted so that widths are proportional to relative likelihood. The rendering of the causal graph, with varying widths of lines between the nodes, resembles a Sankey diagram, or the famous Minard visualisation of Napoleon's march, where line width is proportional to some quantity [31]. In our novel variant of Bayes networks, the width of each connecting path does not represent a measured material flow, but the relative likelihood of that path.

E. "Ghost" Cars

The Monte Carlo simulation of multiple histories is also used to generate a distinctive behaviour in the rendering, in which multiple possible universes are overlaid[2]. For each moving vehicle controlled by an agent, there is uncertainty associated with the position of that vehicle. These uncertainties may result either from decisions made by the agent, or from embodied processes (for example, the driver inaccurately steering around a corner, or pressing the brake or accelerator pedal more or less hard than intended). Based on these uncertainties, the Monte Carlo simulation generates a range of possible locations for the vehicle at any point in time. After sampling over 1000 simulation runs, some locations will be relatively unlikely, while those near the centre of the outcome distribution will be likely. These likelihoods are displayed in the rendering as semi-transparent "ghost" cars, with likely outcomes being more solid, and the unlikely ones more "ghostly". As a result, pressing the play button in the Multiverse Explorer might show an animation in which the ghosts of a given car diverge, perhaps with one ghost turning left and another proceeding straight ahead, and the relative transparency of the two ghosts reflecting their relative likelihood. Fig. 2 illustrates an example of this.

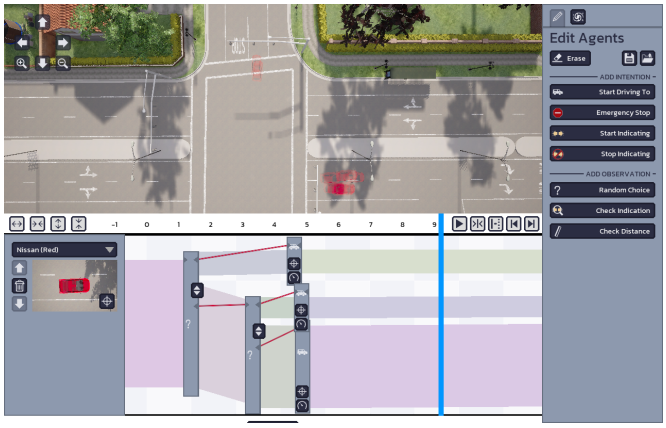


Fig. 2. The partitioned causal graph (timeline) explaining the sequence of decisions being made by a single agent in a simple example. Time is ordered from left to right, and the vertical cursor indicates the point in time at which events in the rendering are currently paused. As 'random choice' nodes are added (corresponding to undetermined or unknown factors in the explanation), the space of possibilities branches out according to prior probabilities of each outcome. The width of the each coloured flow represents the proportion of universes represented by the preceding sequence of events. In this example, the agent has 3 possibilities: driving north, or driving east in one of the two road lanes. The bottom flow is associated to driving east in the southernmost lane, and the prevalence of universes with this possibility is reflected by the resulting opacity of the overlaid ghosts in the rendering. On the right hand side, we have facilities for manually adding explanatory nodes to the timeline - in deployment of the approach these may be policy definitions, automatically generated from AI traces, or investigative hypotheses.

The 'ghost' rendering convention also provides a way to reason about uncertainty in the intentions of other agents. If one agent is expected to act with some prior likelihood, for example whether to turn left or proceed straight ahead, and also conditioned on observations, for example to avoid another car that has turned in front, the Monte Carlo simulation will calculate multiple paths that result from those intentions and observations. These paths are also counted, and ghost cars rendered with transparency depending on the likelihood of a particular course of action. Fig. 3 shows an example where multiple 'ghost' instances of two agents navigate an intersection. Generalisation of the Multiverse Explorer paradigm in the future could use transparency to indicate 'ghostliness' for many kinds of representational or abstract visual elements.

III. Operational Modes

The Multiverse Explorer has two operational modes - one in which alternative policies and expected behaviours of the agents can be explored by modifying the timeline, and one in which the probabilistic space of outcomes generated by

[2]Although our prototype is the first time this visual device has been used to illustrate probabilistic computation, there are some precedents of similar devices in popular culture, such as this sports example to visualise possible ball trajectories resulting from a baseball pitch: https://www.youtube.com/watch?v=jUbAAurrnwU.

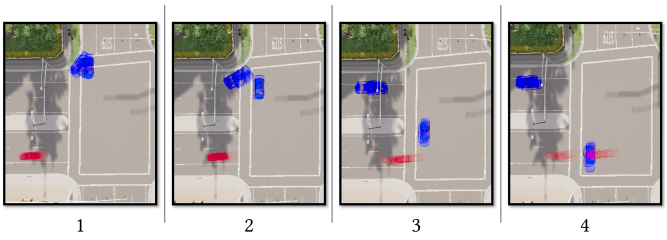


Fig. 3. Four snapshots of the Multiverse Explorer in a simple two agent scenario. The blurry cars represent a superposition of multiple possible states of the agent, where the resulting opacity correlates to the number of observations/universes with agents coinciding in space and time. These states can diverge for reasons such as noise or differences in actions taken by the agents. In this example, the blue car branches out in choosing to turn right or go straight, whilst the red car branches in stopping or following ahead.

the Monte Carlo simulation can be interactively queried in the rendering. We present the two modes using scenarios from the multi-agent driving domain, but generalisation of the Multiverse Explorer paradigm in the future could use the same operational modes for analogous end-user tasks in many domains.

A. Mode 1: Modifying the Timeline

The timeline is designed to represent intentional decisions at a level of granularity comparable to the instructions that might be given to a human driver, for example by a driving instructor or by someone directing a taxi driver to a location that the passenger is familiar with. Rather than low-level vehicle controls such as pressing the accelerator a certain amount or turning the steering wheel to a particular angle, the intention nodes describe an immediate objective, such as to drive toward a particular location, then face a certain direction (as in Fig. 4). The actions required to reach this objective state, including accelerating to the speed limit and turning the steering wheel a certain distance, are all automatically determined using simple Newtonian mechanics. However, each of these mechanical elements has uncertainties associated with it, just as in real driving, such as a steering error. All paths of vehicles are calculated on this basis using discrete timesteps, to make the Monte Carlo simulation tractable.

An agent's actions can be modified directly by editing the timeline using the facilities on the menu shown on the right in Fig. 2. The two kinds of action supported in our initial implementation are either to drive toward a particular position (with the car pointed in a specified direction after arriving there), or to position the car relative to another vehicle. The user can express the intended target location by clicking directly within the rendering (Fig. 4). After the point has been selected, a rotation control is used to rotate the vehicle into the target orientation. The second kind of action is a common situation in city driving - for example overtaking, pulling alongside, or pulling up behind another vehicle. However, the intuitive representation of such intentions is egocentric rather than allocentric, so the target location is rendered from the driver's perspective (Fig. 5). As with the intention to drive toward a certain location, all of the detailed kinematics required to

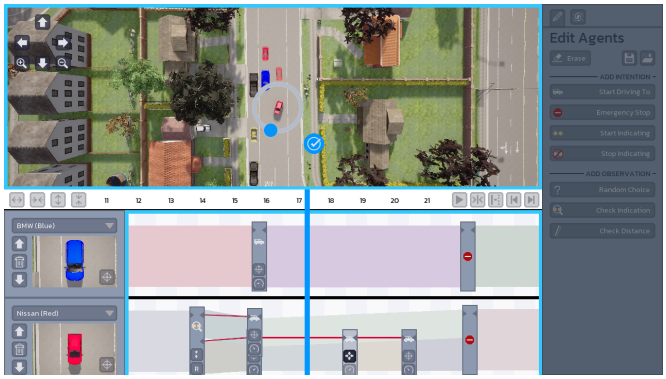


Fig. 4. Representation of an agent's intention to drive toward a certain point on the street, including the desired orientation of the vehicle on arrival, as seen by the circled vehicle with a highlight dot on one side of the circle.

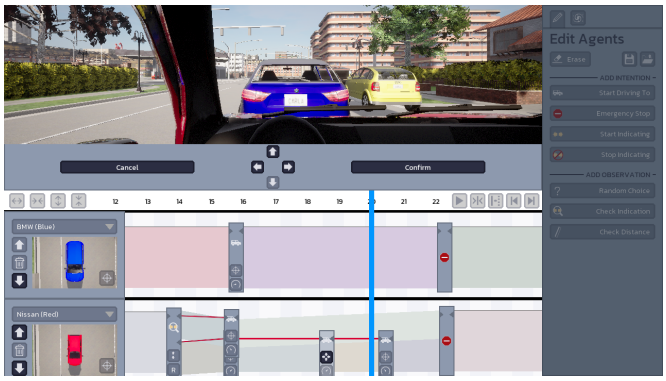


Fig. 5. Representation of an agent's hypothetical intention to position the vehicle relative to another car, using an egocentric rendering of the 3D scene to show how this relative position would look from the perspective of the vehicle's driver.

complete these manoeuvres are calculated automatically, with a degree of uncertainty in the parameters.

In addition to agent action nodes, observation nodes can be added to the timeline. Observation nodes are triggered based on a specified change in the environment, such as an agent reaching a certain position or making a turn indication. Actions that are dependent on what another vehicle is doing may, as in real city driving, be determined by intentional actions of the other driver (such as using a turn indicator to signal intention to turn) or unintentional actions (such as drifting out of their lane or failing to turn a corner properly).

There are a small number of further features that were implemented to support the evaluation scenarios in Section IV, but these relatively straightforward features (e.g. intention to make an emergency stop, or operate turn indicators) are not described in detail here.

B. Mode 2: Filtering Specific Universes

The second operational mode of the Multiverse Explorer allows the user to investigate specific universes by selecting subsets from among the Monte Carlo simulations. To maintain the "multiverse" metaphor, each individual trace from the simulation is described as an alternative universe and given

Fig. 6. Defining query criteria.

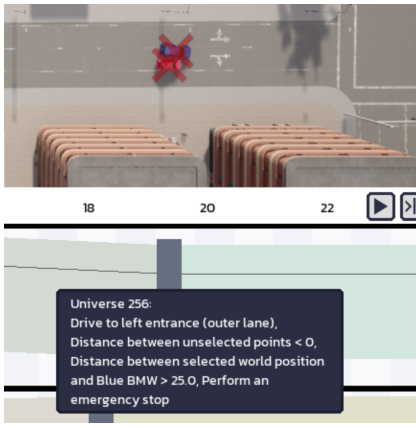


Fig. 7. A filtered universe in which a collision occurs. Note that the vehicles are no longer represented by 'ghosts,' as this is a representation of one particular universe.

an arbitrary display name, such as "universe 375", to reinforce the user's understanding that they are inspecting one of many possible courses of events. Upon selecting a universe, the rendering updates to show only the vehicle positions involved in this specific simulation run, rather than the ghosts of the other multiverse possibilities. Pressing the play button with a single universe selected allows the user to observe the events as they play out in this specific version of the probabilistic scenario.

To aid users in choosing a specific universe to select from the 1000 possible choices, the user can filter all universes according to some criterion that specifies a subset of the possible outcomes, as seen in the modal menu of Fig. 6. This feature can be used, for example, to investigate universes where a specific agent crashes (Fig. 7) or where a specific agent enters a circle placed within the rendering. In scenarios where multiple universes satisfy the given filters, the UI renders all such universes.

IV. User Study

We have evaluated the Multiverse Explorer in a user study designed to investigate whether this style of tool is usable and helpful in understanding basic principles in probabilistic programming. [3]

A. Study Protocol

1) Participants: We recruited 18 participants from our university who had no familiarity with the idea of a multiverse, which was asked as a simple yes/no question prior to scheduling a session. Half the participants (n=9) identified as female and the remainder identified as male. The average age was 22 years. Each participant completed the 40 minute study in-person, in a controlled environment at the university. All participants were briefed on the requirements and risks of participating in the study before consenting. Ethical review and approval was provided by the Cambridge Computer Science ethics committee.

2) Tutorial: Participants were introduced to Multiverse Explorer with a guided tutorial covering key features. Participants observed as the demonstrator added an agent to the road, specified its colour, then added commands to make the agent approach a junction, indicate, check another agent's indicators, and safely make a turn. After checking the participant's understanding of these basic features, the demonstrator explained how the application depicts all possible worlds, with all cars from all worlds shown in the UI.

3) Tasks: Participants were then given three separate tasks and encouraged to think-aloud as they completed them. Task 1 mirrored the tutorial by asking participants to add two agents to the world that drove, indicated, and completed a turn conditioned on the turn indicator of another agent across the intersection. This task only used features that were explained in the tutorial in order to check initial feature understanding and to give participants practice before the more difficult tasks. In Task 2, participants were given a new environment that contained two agents, then randomly assigned a condition, either Rendering or Timeline. Each group was asked to explain what would happen in the environment, but those in the Rendering group (P10-P18, n=9) could only see the top half of the UI, while those in the Timeline group (P1-P9, n=9) could only see the bottom half of the UI. Those in the Rendering group were permitted to select Play to watch the scene play out. Task 3 presented the most complex environment, with several agents driving in a city. All participants were asked to use any available tools in the Multiverse Explorer to identify a universe where the yellow car crashes, then determine why the crash occurs and how the crash could be prevented. All participants could access the entire UI (i.e. neither half was hidden). Additionally, participants were given a tutorial that demonstrated the filter feature for quickly identifying and visualising single universes based on specific criteria, as described in Section III-B.

To conclude the study, participants were invited to share any general thoughts and feedback about the tool.

B. Analysis

Throughout all tasks, the demonstrator documented any notable thoughts that participants shared aloud, as well as scoring a number of tasks for accuracy. Task accuracy included checking participant understanding of all the basic features in Task 1, and if the correct explanation was provided for tasks 2 and 3. Each participant was scored on 10 accuracy checks (Table I), which were each recorded as correct or incorrect by the demonstrator. Half of the participants (P1-P9) received Q7, while the other half (P10-P18) received Q8. This was to account for the conditions assigned in Task 2. After the study, using the notes compiled from participants thinking-aloud as memos, we used open coding to identify themes within the participant responses.

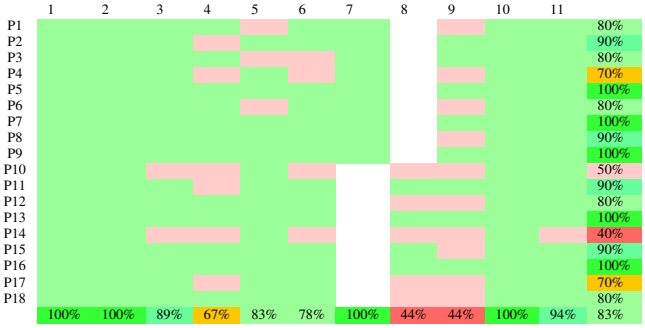
[3]The user study script and tasks are available at https://www.cl.cam.ac.uk/~afb21/publications/multiverse-explorer/

TABLE I
DESCRIPTION OF ACCURACY CHECKS

	Task	**Accuracy Check**[1]
Q1	1	Place car
Q2	1	Change car color
Q3	1	Move car to desired location
Q4	1	Coordinating turn times of two vehicles
Q5	1	Set indicator light before turning
Q6	1	Conditional turn: base turn on indicator of other car
Q7	2(a)	Describe what is occurring using only the timeline
Q8	2(b)	Describe what is occurring using only the rendering
Q9	3	Filter out a universe where the yellow car collides
Q10	3	Describe why the yellow car collides
Q11	3	Describe how to prevent yellow car from colliding

[1] Demonstrator asked each participant to complete the action, then recorded whether the action was executed successfully or not

TABLE II
ACCURACY CHECK STATISTICS



Q7 was only answered by the Timeline group (P1-P9), while Q8 was only answered by the Rendering group (P10-P18).

V. RESULTS

We report our findings about the usability of Multiverse Explorer based on the task accuracy checks and think-aloud utterances of the participants.

A. Task Accuracy

Table II shows the accuracy achieved by each participant on their 10 task accuracy checks, as well as the overall accuracy on each check. Each row represents performance by a participant on all accuracy checks that they answered, and each column represents performance by all participants across a specific accuracy check. Each box represents an outcome; green is correct, and red is incorrect. Some boxes are white, which represents an unanswered accuracy check resulting from the assigned conditions in Task 2. The percentages along the right and bottom side of Table II represent the overall percentage of questions answered correctly in each row and column.

Participants scored an average of 83%, with five participants (28%) achieving a perfect score. Four participants (22%) scored substantially below the average, with one (P14) scoring below 50%. Of the 11 questions asked, four were answered correctly by all participants. The most difficult were Q4 (67% correct), Q8 (44% correct), and Q9 (44% correct).

Participants understood the fundamentals of using Multiverse Explorer, averaging an accuracy of 86% across the six questions in Task 1. The most challenging question was Q4, because it involved multiple elements: coordinating two turning cars, understanding how to create actions, identifying that each vehicle has its own swimlane, and placing the actions in the correct time order. Q6 similarly involved more complexity to coordinate one car observing the turn signal of another car.

Task 2 showed a significant difference (t-statistic = 3.16, p-value = 0.006) in performance between the rendering group (Q7: 100%) and timeline group (Q8: 44%). This discrepancy was due to participants not being able to understand the environment well without the rendering, an idea we explore further below in our analysis of the themes. Task 3 caused considerable confusion at the start, with Q9 (44%) indicating that participants had trouble understanding how to identify and view a single universe using the filtering tool. Once participants managed to apply the filter, either on their own or, if truly stuck, with a small nudge from the demonstrator, they accurately used their skills and experience with the tool to answer Q10 and Q11 with a combined accuracy of 97%.

B. Themes

1) ***Filter Feature Confusion****:* The consensus among the majority of participants was that the filtering feature was confusing. P3, for instance, *"found the filters very confusing."* and went on to point out that *"I want to see a plain English definition that tells me I am filtering all worlds where the yellow agent has collided before 22 seconds. Then I can edit the words 'yellow' and '22' to be 'red' and '30', or whatever I want to find."* P4 shared this sentiment: *"Searching for a particular universe was not intuitive for me, as I was constantly changing the universe by one and then checking if that universe was behaving the way I wanted it to for the task. I think I can use the filters to do this more easily, but they are confusing to use."* P6 was similarly confused by the filters: *"Intuitively, I know that filtering the specific universe I want will help me solve this task, but I don't think I added that filter correctly."* It is clear that the ability to filter universes is a feature that participants would like use in completing common tasks with the Multiverse Explorer, but several participants found this difficult to do in the current UI.

2) ***Ghost Cars Proved Effective****:* Several participants pointed out that the ghost cars added clarity to the idea that the environment depicted all worlds. P12 was particularly excited about this feature: *"For the first time, I can conceptualise the idea of a multiverse. The key was the visualisations - seeing the ghost cars made it click for me that I was seeing all worlds at once."* P17 added: *"The transparent cars are the cars in other universes, right?"* After the demonstrator confirmed this, P17 continued: *"Wow, it's just like racing against the high score in a video game, but it's sort of like you are seeing all possible scores. Makes sense now, really cool."* P1 had a different perspective on the visualisations: *"Are the ghost-cars meant to show me the right way to drive?"* Upon explaining

that they were actually to see all other possible outcomes, P1 replied: *"I see it now, so the ghost cars let me explore the multiverse. That helps me out a lot."* Additionally, once a filter was applied to only depict a single universe, participants related the new view back to the all-worlds view. For instance, P13 noticed that *"this is much different than seeing all the ghost cars. It makes sense though, I'm just looking at only one of the ghosts now, which means this is a single universe within the multiverse."* P4 noted that *"I was a bit confused by all the see-through cars, but now I realise that I was just seeing a bunch of simulated universes all at once, and now I am looking at a single one of these universes."* P17 even continued the video game analogy: *"Now, it's like I'm just watching one of the high scores, not all of them."* It is evident that grasping the concept of a multiverse was facilitated by the juxtaposition of the all-universes view with the filtered, single-universe view.

3) ***Timeline-only View is Difficult****:* For most participants assigned to the timeline-only view in Task 2, Multiverse Explorer became harder to use. P4 exclaimed: *"The timeline still confuses me a little. I like to pair the timeline's story with the visual cues on top to make sure I understand what's going on. This version is much harder."* P8 agreed: *"Can I skip to the next part, this is too hard with just the timeline?"* After being encouraged to try one more time, P8 replied: *"It's like I'm driving the cars blindfolded. I know what the controls do, but I can't see what will happen if I use them."* P7 added that *"Even though the timeline gives me a lot of details, I'm struggling to visualise how this would all play out in the multiverse."* These responses all indicate that the participants struggled to visualise how the timeline translates to actions in the rendering. Additionally, the rendering appears to be the primary tool that participants use to understand the environment, with the timeline serving as the control panel. P5 explained that *"I see the timeline like my video game controller, and the rendering as the screen with the game on it. I can't play the game without the controller, and it can't see the game without the screen. I need both halves of the Multiverse Explorer interface to operate it."*

VI. Design Implications

Based on the results of our user study, we comment below on a few implications that our findings have for the design of future applications that incorporate probabilistic components and/or representations of the multiverse.

A. Visualising Probabilistic Environments

We found strong evidence that access to the rendering when reasoning about the tasks was critical to participant performance. Probabilistic settings can be difficult to visualise, so a rendering of the environment is a key component of Multiverse Explorer and other applications that force users to control and/or interpret probabilistic agents. Furthermore, the ghost-cars were a recurring point of positive feedback from participants during their think-alouds; these visualisations aided participants by giving an omniscient view that served as a starting point from which to filter. Further investigation is needed to extend the ghost-cars example used here to a broader paradigm for probabilistic visualisations, but the idea is promising. For example, this concept could apply to probabilistic samplers by creating a visual representation of the sample space, or even to simple, everyday scenarios like weather forecasting where users could be shown how a human would look as they walk through several simulated worlds where it may be sunny, rainy, snowing, etc.

B. Leveraging Mental Models of the Multiverse

In applications where users need to reason about numerous possible outcomes, an idea we refer to as the multiverse in this work, the interface should align the visualisations with the mental models of the users. For several participants, the "aha" moment for understanding the multiverse occurred when they could connect it back to a previous visualisation, whether internal or external to the Multiverse Explorer. For example, we found that the idea of a multiverse became more intuitive for users once they related it to a past example (e.g. a video game) or even to a previous visualisation within the Multiverse Explorer itself, such as comparing the single-universe view with the all-universes view.

C. Making System State Understandable

There were several instances during the user study where participants struggled to understand the current state of the system. Examples include when some participants did not have access to the rendering in Task 2, when confusion arose when filtering out a single universe, and when some participants could not identify what the ghost cars represented in the all-universes view. In each of these cases, the state of the system could have been more explicit. Remedies for the above UI shortcomings could include having a more detailed timeline view, a more interpretable filtering UI, and having a label that displayed when the rendering toggled between all universe and single universe view. These design elements can be considered in relation to the demands of abstract reasoning that are inherent in any programming task, such that abstract specification of behaviour must be related to concrete instances [3]. In PPLs, where program state is defined in terms of a distribution of possible values rather than a single value for each variable, the abstraction demands are substantially higher. The current version of Multiverse Explorer demonstrates that this cognitive challenge can be mitigated with visualisation, even for non-experts, but further work remains to be done.

VII. Related Work

The basic approach of Multiverse Explorer is a novel paradigm, although the "ghost" style rendering of alternative histories has occasionally appeared in popular culture settings including video games. The informal idea of a multiverse, within which there is a cosmological or ontological relationship between likelihood and causality in alternate universes, has become far better known through the release of the movie Everything Everywhere All at Once, described by IGN

as the 'most awarded movie ever'. (Noting that design and implementation of the Multiverse Explorer was completed in Summer 2021, before the release of that movie).

It is interesting to compare the 3D scene rendering of the Multiverse Explorer, with our inclusion of 'ghost' vehicles, to the rendering style used by Brandao et al [5] to visualise a motion planning path as a sequence of overlaid renderings of the moving robot. In that work, alternative paths are visualised as separate image frames, with one sequence in each frame. Our approach differs in that alternatives are superimposed within the same frame, which offers an advantage in the ability to compare paths (within the the 'multiverse' as we call it), but of course a corresponding trade-off that multiple positions along the path are seen as moving objects over time as the animation plays.

The basic design strategy of a separate block-code editor and world model that are integrated into a single IDE is familiar from educational programming environments such as Alice [11] and Scratch [25]. Continuous execution, with live update of the world model in response to code changes, provides substantial advantages for explorability and understandability, as demonstrated by Tanimoto [29], and subsequently popular in many software engineering environments [8] that draw heavily on the design philosophy of Smalltalk [20].

We are not aware of many interactive approaches that have integrated Monte Carlo simulation into interactive end-user visualisations, with the notable exception of an early GPS navigation application, using particle filters for traffic modelling, that was demonstrated by Williamson et al [32].

Previous work in our group has demonstrated that visualisation of causal graphs can assist with construction and interpretation of programs in a PPL [17], and others have shown that live visualisation of distributions can be integrated into a probabilistic database for data science applications [26]. Following the pioneering work of Erwig and Walkingshaw [15], our group has explored whether interactive visualisation of causal graphs, probability distributions, and Monte Carlo simulations offer educational benefits in the teaching of probability [1], [4].

We are, of course, indebted to the work of the many groups who have been developing and extending the concept of probabilistic programming itself, including but not limited to the teams developing BUGS [27], Stan, [9], PyMC [21], Turing [16], Anglican [30] and Gen [12].

VIII. Future Work

As with the causal graph representation, we have only implemented a minimal set of query interaction features, sufficient to verify that the overall approach is effective, and as a functional basis for an initial user study. There are many opportunities to extend these exploration facilities, drawing on visual query languages and query-by-example systems, as well as interaction techniques from visual constraint editors and other explanation interfaces.

Earlier work in our group showed that interaction with a programmatic representation of a causal network, as used in probabilistic programming languages, becomes an element of usable explanation [1], [17]. Paired with the idea that the complexity of the autonomous vehicle domain calls for explanations in human-agent systems [18], [24], a useful extension of this work would include adding ideas from explainable AI.

We only recruited participants that had no prior experience with the concept of a multiverse, so future studies should compare our results to a study run with users who can already conceptualise the notion of a multiverse. Such a comparison may uncover additional design implications in order for future tooling to address the needs of all users, regardless of domain-knowledge depth.

Finally, an interesting extension of our multi-agent driving case study would be to test whether this demonstrator might be valuable for end-users in the broader domain of transportation. Examples might include systems engineers, urban planners, accident investigators, policymakers, litigators, or even technically knowledgeable end-users curious to find out why their own autonomous vehicle responded the way that it did to a recent unexpected situation. The Multiverse Explorer paradigm might be a useful tool for many classes of user; further research is needed.

IX. Conclusion

We have presented the Multiverse Explorer, a prototype illustrating a novel, domain-specific probabilistic programming language. The core elements of this system are the potential to analyse interaction between multiple heterogeneous agents (including both human and AI agents); support for causal reasoning by integrating models of agent intention with a probabilistic kinematic model; support for counterfactual reasoning within a Bayesian framework by visualising the distribution of outcomes over a Monte Carlo simulation; integration of that simulation with a rendering of a physical world view where alternative courses of action are visualised as 'ghosts'; and a query mechanism that allows the user to highlight and request narrative explanations of causal factors within any one of these simulated universes. As we have noted, the Multiverse Explorer paradigm is potentially applicable to many domains other than multi-agent driving. Nevertheless, the support for temporal and spatial locality of effect seems particularly relevant to those domains where agents interact within a physical and kinematic world model.

This project has contributed to a longer-term research objective, which is to develop domain-specific PPLs (DS-PPLs) that allow some of the principles of probabilistic programming language design to be made more accessible to end-users [2]. These languages have from the outset been inspired by the work of Judea Pearl, who argues persuasively for the importance of diagrammatic causal graphs in order to make probabilistic causal reasoning more accessible to all members of the population [23].

References

[1] Z. Attahiru, R. H. Maudslay, and A. F. Blackwell. Interactive bayesian probability for learning in diverse populations. In *Proceedings of the Psychology of Programming Interest Group (PPIG)*, 2022.

[2] A. Blackwell, T. Kohn, M. Erwig, A. G. Baydin, L. Church, J. Geddes, A. Gordon, M. Gorinova, B. Gram-Hansen, N. Lawrence, et al. Usability of probabilistic programming languages. *Proceedings of the Psychology of Programming Interest Group (PPIG 2019)*, 2019.

[3] A. F. Blackwell. First steps in programming: A rationale for attention investment models. In *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments*, pages 2–10. IEEE, 2002.

[4] A. F. Blackwell, N. J. Bidwell, H. L. Arnold, C. Nqeisji, K. Kunta, and M. M. Ujakpa. Visualising bayesian probability in the kalahari. In *Proceedings of the Psychology of Programming Interest Group (PPIG)*, 2021.

[5] M. Brandao, G. Canal, S. Krivić, P. Luff, and A. Coles. How experts explain motion planner output: a preliminary user-study to inform the design of explainable planners. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pages 299–306. IEEE, 2021.

[6] B. Brown. The social life of autonomous cars. *Computer*, 50(2):92–96, 2017.

[7] B. Brumitt and M. Hebert. Experiments in autonomous driving with concurrent goals and multiple vehicles. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 3, pages 1895–1902. IEEE, 1998.

[8] B. Burg, A. Kuhn, and C. Parnin. 1st international workshop on live programming (live 2013). In *2013 35th International Conference on Software Engineering (ICSE)*, pages 1529–1530. IEEE, 2013.

[9] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76, 2017.

[10] O. Carsten and M. H. Martens. How can humans understand their automated cars? hmi principles, problems and solutions. *Cognition, Technology & Work*, 21(1):3–20, 2019.

[11] S. Cooper, W. Dann, and R. Pausch. Alice: a 3-d tool for introductory programming concepts. *Journal of computing sciences in colleges*, 15(5):107–116, 2000.

[12] M. F. Cusumano-Towner, F. A. Saad, A. K. Lew, and V. K. Mansinghka. Gen: a general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th acm sigplan conference on programming language design and implementation*, pages 221–236, 2019.

[13] L. De Raedt, R. Manhaeve, S. Dumancic, T. Demeester, and A. Kimmig. Neuro-symbolic= neural+ logical+ probabilistic. In *NeSy'19@ IJCAI, the 14th International Workshop on Neural-Symbolic Learning and Reasoning*, 2019.

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[15] M. Erwig and E. Walkingshaw. A visual language for explaining probabilistic reasoning. *Journal of Visual Languages & Computing*, 24(2):88–109, 2013.

[16] H. Ge, K. Xu, and Z. Ghahramani. Turing: a language for flexible probabilistic inference. In *International conference on artificial intelligence and statistics*, pages 1682–1690. PMLR, 2018.

[17] M. I. Gorinova, A. Sarkar, A. F. Blackwell, and D. Syme. A live, multiple-representation probabilistic programming environment for novices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2533–2537, 2016.

[18] L. Guerdan, A. Raymond, and H. Gunes. Toward Affective XAI: Facial Affect Analysis for Understanding Explainable Human-AI Interactions. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 3789–3798, 2021.

[19] R. L. Harrison. Introduction to monte carlo simulation. In *AIP conference proceedings*, volume 1204, pages 17–21. American Institute of Physics, 2010.

[20] A. C. Kay. The early history of smalltalk. In *History of programming languages—II*, pages 511–598. 1996.

[21] A. Patil, D. Huard, and C. J. Fonnesbeck. Pymc: Bayesian stochastic modelling in python. *Journal of statistical software*, 35(4):1, 2010.

[22] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288, 1986.

[23] J. Pearl and D. Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

[24] A. Raymond, H. Gunes, and A. Prorok. Culture-Based Explainable Human-Agent Deconfliction. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pages 1107–1115, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems.

[25] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[26] F. Saad and V. K. Mansinghka. A probabilistic programming approach to probabilistic data analysis. *Advances in Neural Information Processing Systems*, 29, 2016.

[27] D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. Bugs 0.5 examples volume 1 (version i). *MRC Biostatistics Unit, Institute of Public Health*, 1996.

[28] S. Sun, C. Zhang, and G. Yu. A bayesian network approach to traffic flow forecasting. *IEEE Transactions on intelligent transportation systems*, 7(1):124–132, 2006.

[29] S. L. Tanimoto. Viva: A visual language for image processing. *Journal of Visual Languages & Computing*, 1(2):127–139, 1990.

[30] D. Tolpin, J.-W. van de Meent, H. Yang, and F. Wood. Design and implementation of probabilistic programming language anglican. In *Proceedings of the 28th Symposium on the Implementation and Application of Functional programming Languages*, pages 1–12, 2016.

[31] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[32] J. Williamson, S. Strachan, and R. Murray-Smith. It's a long way to monte carlo: probabilistic display in gps navigation. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 89–96, 2006.