Probabilistic Logic Programming Semantics for Procedural Content Generation

Abdelrahman Madkour, Chris Martens, Steven Holtzen, Casper Harteveld and Stacy Marsella

Northeastern University

Abstract

Research in procedural content generation (PCG) has recently heralded two major methodologies: machine learning (PCGML) and declarative programming. The former shows promise by automating the specification of quality criteria through latent patterns in data, while the latter offers significant advantages for authorial control. In this paper, we propose the use of probabilistic logic as a unifying framework that combines the benefits of both methodologies. We propose a Bayesian formalization of content generators as probability distributions and show how common PCG tasks map naturally to operations on the distribution. Further, through a series of experiments with maze generation, we demonstrate how probabilistic logic semantics allows us to leverage the authorial control of declarative programming and the flexibility of learning from data.

Why Bayesian reasoning for PCG?

To be able to allow designers to come up with complex generators that remain controllable and understandable, we argue that we need a solid framework for dealing with randomness. In every formulation of PCG, we define a possibility space and draw instances from it with some level of randomness. The degree to which this randomness is specified is often implicit in the techniques themselves The effect of doing so is obfuscation of the way randomness is controlled to other aspects of the generator specification, which in turn makes it difficult to reason about the generator. Diversity is an important component of many PCG systems, and achieving diversity requires a certain degree of randomness. A main challenge in developing tools for PCG is empowering designers to define and control this randomness. Bayesian reasoning affords us with formal mechanisms we require to reason about randomness.

Probabilistic Logic Programming (PLP)

PLP is a programming paradigm that combines the declarative features of logic programming and the Bayesian semantics of probabilistic programming. Many PLP formalisms exist but for illustration purpose we consider Problog [1]. Below is a code snippet in Problog that generates a 5x5 maze.

```
0.5::edge(1,2,0);0.5::edge(1,2,1).
...
0.5::edge(24,25,0); 0.5::edge(24,25,1).
edge(X,Y,V) :- edge(Y,X,V).
path(X,Y) :- edge(X,Y,1).
path(X,Y) :- edge(X,Z,1), Y\=Z, path(Z,Y).
evidence(path(1,2)).
...
evidence(path(1,25)).
query(edge(_,_,1)).
```

In addition to Prolog's declarative specification of graph reachability we have annotated facts, such as $0.5 :: edge(1, 2, 0)$, which allows us to encode the probability distribution over the possible mazes.

Types of constraints in this formulation

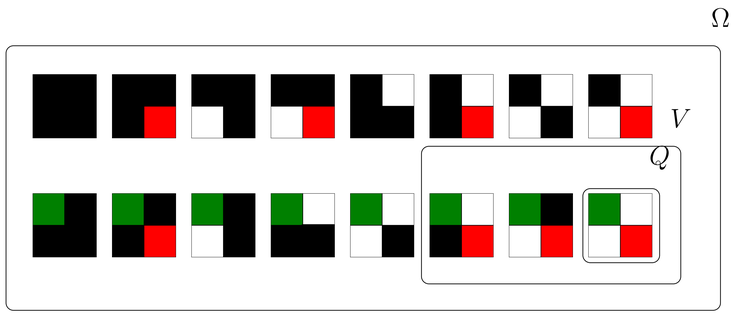
We formulate the desiderata on generated content in terms of constraints. Under a Bayesian formulation we define the space of all possible content in a given generative task as the *generative space* and condition on constraints to arrive at a desired generator.

- **Validity Constraints:** Hard constraints that we impose on artifacts to ensure they are usable.
- **Quality Constraints:** A mixture of hard and soft constraints that constitute design intentions over valid content. They can do this by either steering the generator away from certain outcomes or eliminate them entirely.

The distinction is a notation convention, as any hard quality constraint can be folded into validity constraints. However, this distinction emphasizes the disparate goals of each of the types constraints and allow a generator under certain validity constraints to be reused with different sets of quality constraints.

Grid Example

For illustration purposes let's consider generating a 2x2 grid based level, where the objective is to get from the top left(green tile) to the bottom right(red tile), and we might have untraversable (black) tiles. A valid grid is one where there is a path from top left to bottom right, and a quality grid is one with multiple such paths.



A Bayesian formalism for PCG

Let Q be the set of artifacts consistent with a set of quality constraints F, $P(Q)$ be the probability that a generator P generates Q, where Q contains assignments to the virtual evidence defined by F, V be a set of valid artifacts that P can generate and $P(V)$ the probability that P generates V. Then a *quality generator* $P(V|Q)$, is defined:

$$P(V|Q) = \frac{P(Q|V)P(V)}{P(Q)}$$

where $P(Q|V)$ is the probability of generating quality artifacts given that the generator generated valid artifacts V.

Maze Generation: Setup

To show some of the concrete benefits of the probabilistic approach we ran a small experiment generating 1000 5x5 mazes using Answer Set Programming and Problog. We then compute metrics proposed by [2] and visualize the distribution over these metrics. We further tested the learning capabilities of Problog and show that even after learning the generator remains diverse.

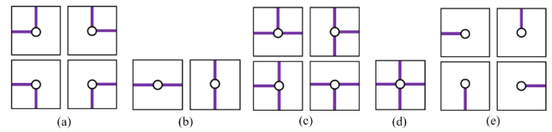
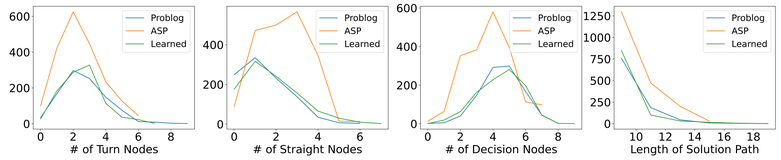


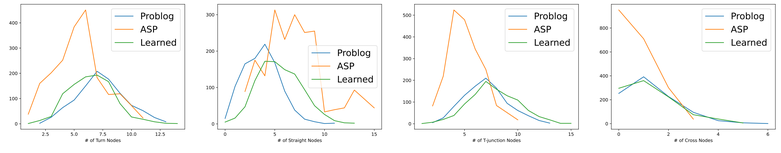
Figure 1. Node types used in the metrics proposed by kimDesigncentricMazeGeneration2019. **(a)** shows **Turn nodes**, **(b)** shows **Straight nodes**, **(c)** shows **T-junction nodes**, **(d)** shows a **Cross-Junction** node and **(e)** shows **Terminal nodes**. Image from [2].

Maze Generation: Results

Our experiment demonstrates how an explicitly probabilistic approach allows us to define a generator that is diverse in designer relevant metrics.



Counts of solution path metrics proposed by [2] for ASP, Problog and Learned Problog generated mazes



Counts of maze metrics proposed by [2] for ASP, Problog and Learned Problog generated mazes

References

[1] Anton Dries. Declarative Data Generation with ProbLog. In *Proceedings of the Sixth International Symposium on Information and Communication Technology*, pages 17–24, Hue City Viet Nam, December 2015. ACM.

[2] Paul Hyunjin Kim, Jacob Grove, Skylar Wurster, and Roger Crawfis. Design-centric maze generation. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, pages 1–9, San Luis Obispo California USA, August 2019. ACM.