Final Research Report:

Analyzing Computer Randomness: BLOG vc. C++

Amanda Wolski

Yelizaveta Semikina

Advisor: Professor Hummel

Early Research Scholars Program (ERSP)

May 6, 2021

1. Introduction and Research Context

Probabilistic programming is a style of programming used to solve complex probabilistic models, such as those with a large number of variables, or those where more data becomes available over the course of the problem. This type of programming may be very useful for certain types of artificial intelligence (AI). For instance, a self-driving car must keep track of many factors (e.g. its position, the location of other cars on the road, GPS location, traffic signs) and use those factors to make quick decisions. Aside from artificial intelligence, probabilistic programming can also be applied towards financial markets by defining the price forecast or predicting trades. More and more probabilistic programming languages are constantly being built for specialized purposes.

We tested one of these languages to see its efficiency at solving simple probabilistic problems compared to C++, a popular programming language that is used for game development, database applications, operating systems, etc. Through this process, we also develop a deeper understanding of probabilistic programming and its usage; more specifically, we have chosen BLOG as the probabilistic programming language due to its readily available documentation.

2. Problem and Background

Despite probabilistic programming languages growing for financial markets, the criminal system, and specialized functionality in artificial intelligence, it still remains a niche topic in computer science. Therefore, the motivation behind this research was to determine how much more efficient it is to utilize a probabilistic programming language, such as BLOG, over C++. Hopefully, the results of our research will peak the interests of other undergraduate students and computer scientists that are uncertain whether a probabilistic programming language could benefit their project.

We programmed probabilistic programming concepts in C++ and in BLOG. We decided to choose C++ over other programming languages, because the people in our group have had the most experience with this language at UIC. Then we attempted to write the C++ programs in BLOG and the BLOG programs in C++. Afterwards, we planned to compare the programs on length, runtime, effort, and effectiveness. Based on these factors, we determined whether BLOG performs as expected. In this case, "as expected" would mean making the probabilistic problems easier and faster to solve. This would answer the question of, how truly random is C++? Is BLOG a viable (or even better) substitute in probabilistic programs? Also, we determined whether BLOG can be used for non-probabilistic problems and how useful it is. Figure 1 utilizes a flowchart to compare a typical computer programming language to the process of modeling with a probabilistic program, to highlight why C++ and BLOG have been chosen in this research.

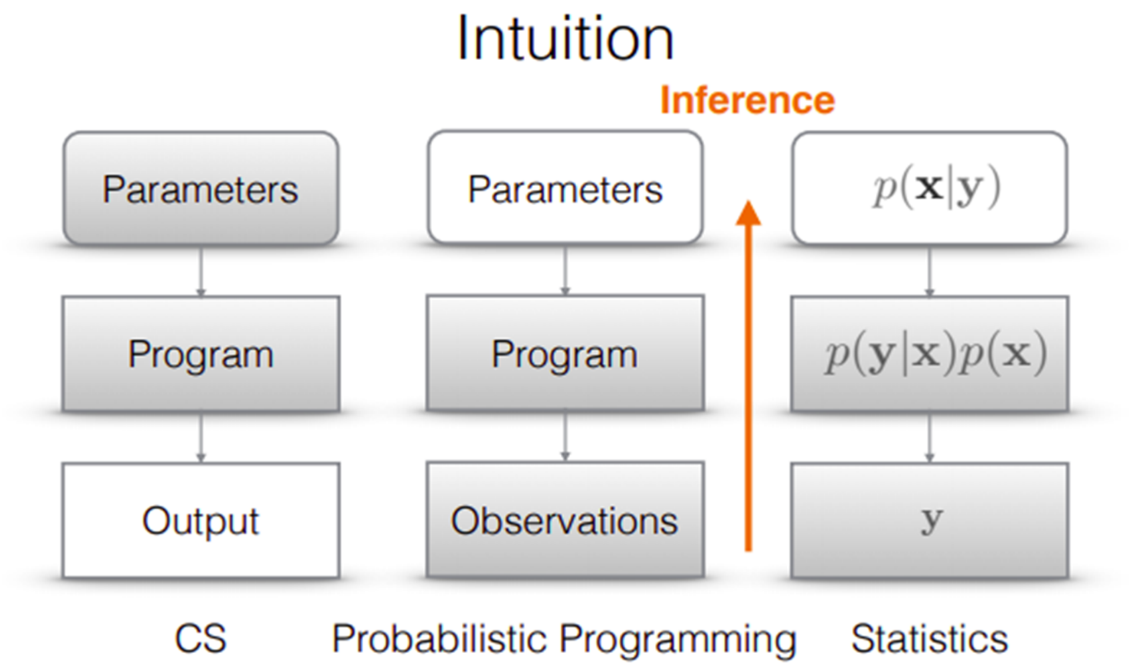


Figure 1: *Comparing a typical computer programming pipeline (left side) vs. the process of modeling in statistics (middle and right side).*

3. Methodology and Approach

How did we compare BLOG and C++ for probabilistic programming?

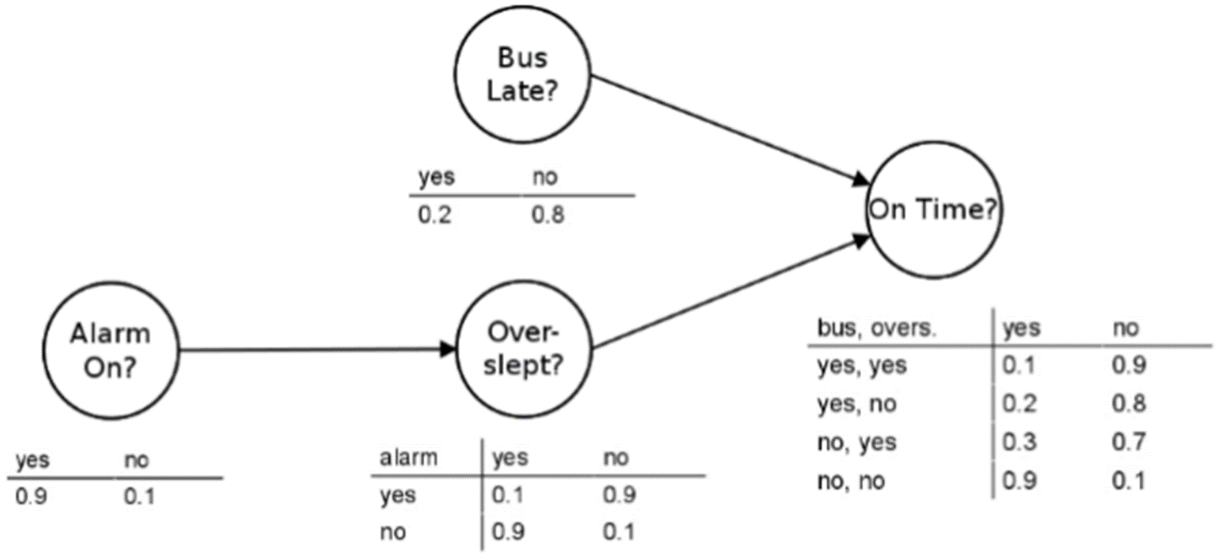
- We learned about and familiarized ourselves with BLOG.
- We focused on two simulations during our research: a Coin Flip Simulation, and a Slot Machine Simulation.
- We utilized queries and compared how observational randomized data in BLOG differs from the rand() / srand() functions in C++. We used Figure 3 as an example for our research to answer questions such as: How does the probability of a head/tail change as we continue to flip the coin? How does the probability of winning in a slot machine change as we continue to take turns? How does my chance of winning change?

Images 1 & 2: *Slot machine (left), coin flip (right).*

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

A, B	= events
$P(A\|B)$	= probability of A given B is true
$P(B\|A)$	= probability of B given A is true
$P(A), P(B)$	= the independent probabilities of A and B

Figure 2: *Bayes' rule determines the probability of event(s), based on the knowledge of previous conditions (or previous probability) related to the event(s).*



yes	no
0.2	0.8

yes	no
0.9	0.1

alarm	yes	no
yes	0.1	0.9
no	0.9	0.1

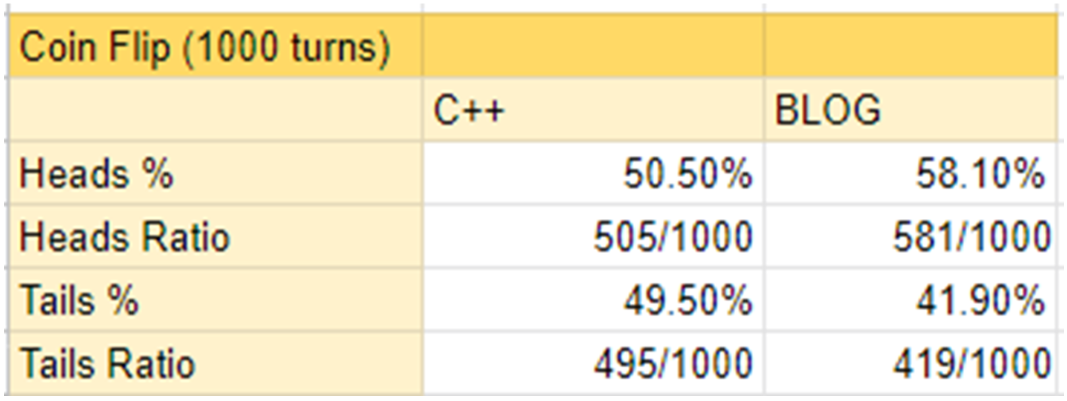
bus, overs.	yes	no
yes, yes	0.1	0.9
yes, no	0.2	0.8
no, yes	0.3	0.7
no, no	0.9	0.1

Figure 3: *"Student's Day" example of the query model used in the Coin Flip Simulation, where previous events impact the outcome of the next event.*

4. Results

Coin Flip Simulation (Figure 4):

- In the C++ Coin Flip Simulation, we wrote a void function called coin_flip() which generates a random number between 1-2 using rand() and prints the output. 1 represents head and 2 represents tails. In the main function, we call coin_flip() based on how many times the user wants to use a for loop.
- The BLOG Simulation was shorter in length and took less time to finish running.
- The BLOG simulation also utilized queries which provided output of heads or tails based on previous observations. The C++ program does not have this feature, making the BLOG data more accurate to an authentic coin toss.

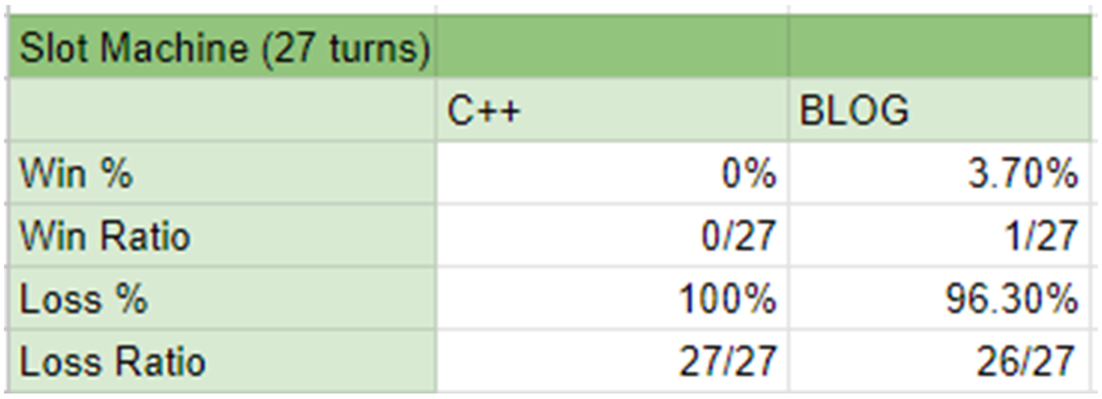


Coin Flip (1000 turns)		
	C++	BLOG
Heads %	50.50%	58.10%
Heads Ratio	505/1000	581/1000
Tails %	49.50%	41.90%
Tails Ratio	495/1000	419/1000

Figure 4: *Table representing the data from Coin Flip Simulations in C++ and BLOG. Includes the heads/tails percentage and ratios.*

<u>Slot MachineSimulation</u> (Figure 5):

- In the C++ Slot Machine Simulation, we wrote a class called slot_machine. Every object of slot_machine contains 3 ints: image1, image2, and image3. They represent each knob on the slot machine. In our case, we have 3 images on each knob. The void spin() function uses rand() to select a number 0-2 for each image variable. To win, the user must get 0 0 0, 1 1 1, or 2 2 2.
- The BLOG simulation was also shorter in length and took less time to finish running.
- When calculating the probability of winning using Figure 2, we would take (1/3)*(1/3)*(1/3) which is 1/27. Even after 27 turns, the C++ simulation could not produce a win because the output was not truly randomized every turn.



Slot Machine (27 turns)		
	C++	BLOG
Win %	0%	3.70%
Win Ratio	0/27	1/27
Loss %	100%	96.30%
Loss Ratio	27/27	26/27

Figure 5: *Table representing the data from Slot Machine Simulations in C++ and BLOG. Includes the wins/losses percentage and ratios.*

5. Conclusion

Based on our results in the previous section, we have concluded that BLOG makes coding probabilistic programs easier and more effective. Although C++ has randomizing functions, they are not completely random because they contain a seed which generates the same sequence each time the code is executed. This happened in the Slot Machine Simulation, where the outcome was not truly randomized, causing 0% wins and 100% losses.

Additionally, there are some functionalities in BLOG that are not readily accessible in C++. For instance, BLOG provides a query language which allows the programmer to ask questions about what will happen next based on previous observations. We utilized this feature in the Coin Flip Simulation, where previous turns impacted the outcome of the next for 1000 turns. Therefore, BLOG has proven that it can be useful for bigger probabilistic projects that manipulate larger sets of data.

6. Acknowledgements

We would like to thank:

- Dr. Renata Revelo for the opportunity to do undergraduate research through the Early Research Scholars Program (ERSP).
- Dr. Joseph Hummel and Marius Zavistanvicius for mentorship.
- Dr. Piotr Gmytrasiewicz and Sarit Adhikari for guiding us to learn BLOG and providing resources.

7. References

[1] ACM SIGPLAN, "Probabilistic Programming: The What, Why and How," *YouTube*, Nov. 16, 2020. Available: https://www.youtube.com/watch?v=cvD9DnTDxmY.

[2] “BLOG,” *BLOG Programming Language*, https://bayesianlogic.github.io/.

[3] Clemente, Fabiana. “Intro to Probabilistic Programming,” *Towards Data Science*, Jul. 7, 2020. https://towardsdatascience.com/intro-to-probabilistic-programming-b47c4e926ec5.

[4] Gordon, Andrew, Henzinger, Tomas. “Probabilistic Programming,” https://www.cs.cornell.edu/courses/cs4110/2016fa/lectures/lecture33.html.

[5] Pachhai, Siddhartha. “Probabilistic Programming Journal 1: Modeling even change,” *Towards Data Science*, Apr. 20, 2019. https://towardsdatascience.com/probabilistic-programming-journal-1-modeling-event-change-9e9a91a5283a.

[6] Parviainen, Pekka. “Bayesian Networks,” *Machine Learning*, Oct. 18, 2019. https://www.uib.no/en/rg/ml/119695/bayesian-networks.

[7] Salvatier, John, et al. “Probabilistic Programming in Python Using Pymc3.” PeerJ Computer Science, vol. 2, 2016, https://doi.org/10.7717/peerj-cs.55.