RH RQ

Bringing great research ideas into open source communities

Kate Saenko

minimizing dataset bias in AI

Sequential Monte Carlo

Efficiently verifying Linux behavior

+

Adaptive learning

Teaching teachers

Changing the world, one lesson at a time

Red Hat Research Quarterly
Volume 2.4 | February 2021 | ISSN 2691-5251

THE UNIVERSAL AI SYSTEM FOR HIGHER EDUCATION AND RESEARCH

NVIDIA DGX A100

Higher education and research institutions are the pioneers of innovation, entrusted to train future academics, faculty, and researchers on emerging technologies like AI, data analytics, scientific simulation, and visualization. These technologies require powerful compute infrastructure, enabling the fastest time to scientific exploration and insights. NVIDIA® DGX™ A100 unifies all workloads with top performance, simplifies infrastructure deployment, delivers cost savings, and equips the next generation with a powerful, state-of-the art GPU infrastructure.

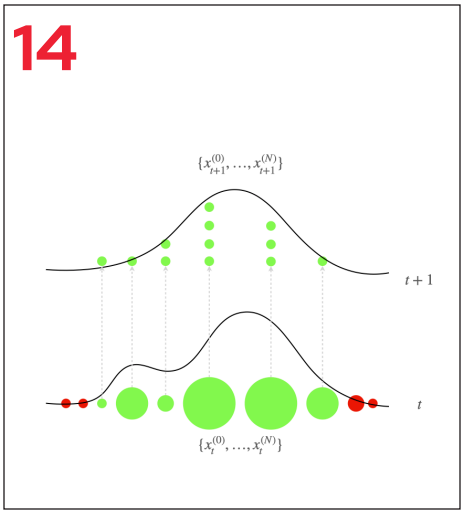
Learn More About **DGX** @ nvda.ws/dgx-pod
Learn More About **DGX on OpenShift** @ nvda.ws/dgx-openshift

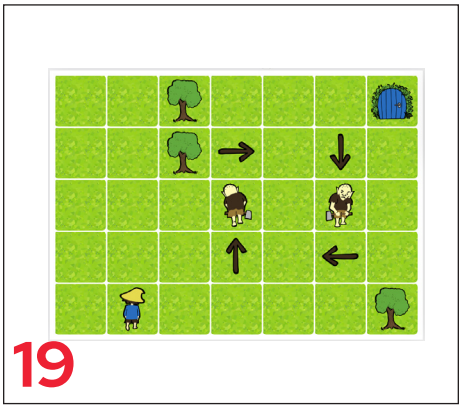
RESEARCH QUARTERLY

VOLUME 2:4

Red Hat

Table of Contents





Departments

Features

NORTH AMERICA
1 888 REDHAT1

EUROPE, MIDDLE EAST, AND AFRICA
00800 7334 2835
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
info-latam@redhat.com

facebook.com/redhatinc
@redhatnews
linkedin.com/company/red-hat

From the Director

Managing large-scale systems

by Hugh Brock

About the Author

Hugh Brock is the Research Director for Red Hat, coordinating Red Hat research and collaboration with universities, governments, and industry worldwide. A Red Hatter since 2002, Hugh brings intimate knowledge of the complex relationship between upstream projects and shippable products to the task of finding research to bring into the open source world.

I have been spending a lot of time lately thinking about all the hard problems involved in managing large-scale systems. Why? Well, it turns out to be a really important topic for Red Hat Research and for the Red Hat engineering community that we hope to serve. If we are correct that operating large-scale systems will necessarily be the domain of "expert systems" with AI, then we need to understand exactly what we mean by "operating," at a minimum.

I tend to approach these kinds of issues from a typical engineering standpoint. How can I construct the "plumbing" that allows me to get decent data out of a system, in the form of logs, events, metering, and so on? And how can I then add the appropriate controls that let someone or something in possession of that decent data do something useful with it? Unfortunately, as hard as this problem is, it turns out to be just the tip of the iceberg. Sanjay Arora's interview with computer vision expert Kate Saenko, our cover story for this issue, focuses on the difficulty of training models, neural networks, and the like so that they are generalizable and not "biased"—biased in the sense that they are unable to tell that an orange hanging from a tree in sunlight is the same object as one sitting in a bowl of fruit in candlelight. This lack of generality also affects the AI we train to control systems. It will be very interesting to see what needs to happen over time before we can really trust a robot to know which tuning knob to turn to keep a mission-critical compute cluster running.

A related problem with large-scale systems arises simply because of the quantity of data they generate and the expense of moving all those bits around. For any reasonably large system, some degree of processing will need to take place close to where the data is collected, so that a smaller amount can be sent on to a central processor. Red Hatter Rui Vieira's article on using Bayesian inference on streaming data is a very deep look at the different methods available to approximate and reduce a very large data flow. I hope to see applications of his work soon.

In addition to training models, we spend a lot of time in this issue on the different ways we train human beings. Check out Tomaš Effenberger's piece on using microworlds and puzzles to teach kids programming—it's absolutely fascinating (and almost certainly more effective than the Fortran books I read at age 12). We don't stop with kids, either. Petr Viktorin writes in this issue about establishing a Python training program for adult women. Through the program he developed, Petr helped a lot of people understand programming, and in return learned a lot from them about agency and motivation. Like children, adults have lots of different reasons to learn. Fortunately, both children and adults learn better than machines—for now, at least. RH RQ

What to expect from Devconf.cz 2021

by Gordon Haff

Like so many other events this year, DevConf.cz is going virtual from February 18-20. Originally an internal Red Hat event held in Brno in the Czech Republic, this free, volunteer-organized event is now in its thirteenth year and is open to all. As usual, DevConf.cz will cover a wide range of topics of interest to *Red Hat Research Quarterly* (*RHRQ*) readers, including considerable representation from academia with talks on researchers' latest projects and findings.

Martin Ukrop and Pavol Žáčik will provide an update on security certificate usability research that is part of the academic cooperation between Red Hat Engineering and Masaryk University. Ukrop wrote about their earlier work on this problem, drawn from research conducted in part at a prior DevConf, in a previous issue of *RHRQ* (see issue 2:2, "Don't blame the developers").

Attila Lakatos is a Red Hat intern in Brno; together with Zoltan Fridrich, he will talk about a software framework that protects your system against rogue USB devices. Another intern, Ondřej Míchal, will cover Toolbox, a tool for creating an interactive container-based environment. Engineering interns are considered part of the Red Hat Research Program, which organizes their mentorship.

A panel discussing the need to realign analysis tools to better address the specifics of modern distributed web apps will provide another spotlight on research projects. The panel will introduce the open source Prophet project, which uses holistic code analysis to better understand the behavior of enterprise microservice apps. Andrew Walker, a PhD student at Baylor University, and Tomas Cerny, a professor there, will be among those on the panel. For an overview of Prophet, go to github: https://github.com/cloudhubs

These examples just scratch the surface of what will be on offer at DevConf. With tracks covering everything from data engineering and data science, to technical documentation and open source community, to all things Linux kernel to containers to storage to quality engineering (and more!), there's something for everyone. Don't miss it.

About the Author

Gordon Haff is Technology Evangelist at Red Hat, where he works on emerging technology product strategy, writes about tech trends and their business impact, and is a frequent speaker at customer and industry events. His books include *How Open Source Ate Software,* and his podcast, in which he interviews industry experts, is Innovate @ Open.

News

Red Hat Research Days 2020– What are we thinking about now?

Highlights from the distributed workflows and infrastructure software tracks

by Gordon Haff

Last September, a quartet of virtual Red Hat Research Days dove into kernel/hardware development, distributed workflows, privacy, and infrastructure software. The virtual conferences also covered some of the research on display in *Red Hat Research Quarterly,* including James Honaker and Merce Crosas' work on balancing data sharing and privacy in a recent issue (see issue 2:2, "Voyage into the open dataverse").

Here, we highlight some of the specific research covered on the distributed workflows and infrastructure software days.

TRAINING AI ON SMALL AND BIASED DATASETS

In an ideal world, machine learning/artificial intelligence (ML/AI) would be able to train using lots of perfectly representative and well-labeled data. In practice, things are rarely that neat and tidy. Kate Saenko, an associate professor at Boston University, studies the very common situation in which models need to learn from biased and small datasets. "I would challenge you to find me a dataset that *isn't* biased," Saenko said.

In an ideal world, machine learning/artificial intelligence would be able to train using lots of perfectly representative and well-labeled data. In practice, things are rarely that neat and tidy.

For example, Saenko suggested, imagine you have a model that's been trained using supervised learning with labeled data to recognize pedestrians in a warm climate. Now, try to use that model in New England in winter. Pedestrians are wearing hats and heavy coats. They may be hidden by snow banks. Even if it's a relatively large dataset, it's biased towards people in a subset of possible environments.

Saenko and her fellow researchers have primarily applied a technique called adversarial domain alignment to improve classification accuracy when presented with new unlabeled data that's not representative of the original. They add a new classifier called a domain discriminator to align the two different datasets without requiring new labeled data.

PROGRAMMABLE NETWORK-CENTRIC INFRASTRUCTURE FOR RESEARCH

The distributed workflows day also featured a

talk by Ilya Baldin, Director, Network Research and Infrastructure, at RENCI (Renaissance Computing Institute). The subject was FABRIC: an adaptive programmable research infrastructure for computer science and science applications. FABRIC is intended to enable cutting edge and exploratory research at scale in networking, cybersecurity, distributed computing and storage systems, machine learning, and science applications.

There are a number of motivations behind FABRIC. In particular, Baldin foresees that changes in the economics of compute and storage allow for the possibility that a future internet might be more stateful. As Baldin put it, "if we had to build a router from scratch today, it wouldn't look like the routers that we build now." Add to this the explosion of new types of compute, like GPUs and FPGAs, a new high-speed intelligent network edge, and new classes of distributed applications. FABRIC should provide new ways to link all these things together.

FABRIC launched in 2019 with a $20 million grant from the National Science Foundation. It's since been expanded worldwide with a sister project called FABRIC Across Borders (FAB) which will link FABRIC's nationwide infrastructure with nodes in other countries. It's intended to give researchers a testbed with network-resident capabilities to explore and anticipate how large quantities of data will be handled and shared among collaborators spanning continents.

As Baldin put it, "if we had to build a router from scratch today, it wouldn't look like the routers that we build now."

FINDING SOFTWARE BUGS MORE EFFICIENTLY

The infrastructure software day led off with a talk by Baishakhi Ray, an assistant professor at Columbia University, on using neural networks to make fuzzing more efficient. A common technique for finding software vulnerabilities, fuzzing is a software testing technique that provides invalid, unexpected, or random data inputs to a program to see if it will crash or otherwise display anomalous behavior.

However, the success of fuzzers can depend on a lot of human judgement, because just using traditional fuzzing techniques can be very inefficient. Take for example the evolutionary techniques that allow the fuzzer to use feedback from each test case to learn the format of the input over time. Even with this relatively advanced technique, the fuzzer can still get stuck in fruitless sequences of random mutations.

Ray's research proposes a novel smoothing technique using neural network models that can incrementally learn smooth approximations of a complex, real-world program's branching behaviors. Evaluations of this technique suggest it not only performs faster than existing fuzzer approaches but can also find bugs that other fuzzers don't.

HOW TO MOST EFFICIENTLY SCHEDULE MULTICORE SYSTEMS?

Finally, a talk by Professor Mor Harchol-Balter and PhD student Benjamin Berg of Carnegie Mellon University also looked at performance, but in the context of scheduling on multicore systems.

On such systems, you can dynamically allocate resources to a given job. But should you give each job a lot of resources so it finishes quickly, or should you be more fair about it and assign fewer resources to more jobs? The answer, it turns out, is that it depends on the nature of a given job. In general, most jobs scale less than linearly so using four cores rather than one gives you less than a 4x speedup. But give every job the bare minimum and, while that may be efficient, everything may take a long time to complete—even jobs that could otherwise complete quickly.

Harchol-Balter and Berg's research is focused on deriving an optimal allocation policy that minimizes mean response time across a set of jobs by balancing the trade-off between granting priority to short jobs and maintaining the overall efficiency of the system.

The recordings of all the sessions are up on research.redhat.com, and we encourage you to give them a view. RH RQ

Interview

When good models go bad: Minimizing dataset bias In AI

by Sanjay Arora

*Sanjay Arora is a data scientist at Red Hat and a member of the Greater Boston Research Interest Group with particular interests in AI and machine learning. For RHRQ he interviewed **Kate Saenko**, a faculty member at Boston University and consulting professor for the MIT-IBM Watson AI Lab, about managing bias in machine learning datasets and the problems that remain unsolved.*

About the Author

Sanjay Arora works at Red Hat's AI Center of Excellence and is mainly interested in the application of machine learning to low-level systems.

Sanjay Arora: At Red Hat Research Day you talked about your research into bias in AI, bias in datasets, and how models can go wrong, especially in computer vision. Could you summarize your thesis or the core idea behind that thrust of your research?

Kate Saenko: Modern AI techniques use machine learning, which means that you can't develop an algorithm without a set of data examples to learn from. Every single AI algorithm these days has been trained on a dataset composed of examples of the kind of inputs that algorithm will receive. For example, if the algorithm is supposed to detect pedestrians, the inputs would be images and the correct outputs that it should be predicting. In this case, if there is a pedestrian in the region of the image then it should predict number one, meaning "I detected the pedestrian," or number zero, meaning "I didn't."

But because every single algorithm has been trained on a fixed-size, finite dataset—it could be 10,000 images of pedestrians or 5,000 images of pedestrians—they're always going to have some sort of bias. This is because there are so many axes of variation in the world, and in particular in the visual world. There could be different lighting, there could be different seasons, different clothing that people are wearing, or even different numbers of people on the street. It couinld be very crowded or there could be very few people very far apart. Your algorithm will inevitably be biased to a certain visual appearance of pedestrians that the dataset presented for training.

Sanjay Arora: In terms of bias, most people think of bias as predisposition, the way we use it in English in general. But in terms of this research, would you say that at least one technical definition of bias would be a distributional mismatch between your training or source distribution, and your inference or target distribution?

Kate Saenko: So bias is a very broad term. And in fact, it's not just datasets that can lead to bias in AI algorithms. AI algorithms themselves could be biased in some way that does not depend on the data. They could even take a dataset that is relatively, shall we say fair, but then the algorithm itself could amplify the bias.

> "There's more being proposed all the time because we are always sort of trying to solve a smaller problem, because the global problem is very, very hard to solve."
> *-Kate Saenko*

As far as technical definitions, what we're talking about here is actually dataset bias. And there are lots and lots of different technical definitions even in my field of research. Domain adaptation is the technique that we're using to overcome dataset bias. There's more being proposed all the time because we are always sort of trying to solve a smaller problem, because the global problem is very, very hard to solve.

So we're always trying to carve out a little piece and say, "Okay, we're going to define this problem this way, with these assumptions, and try to solve it."

Sanjay Arora: When you say algorithmic bias, do you mean things like an inductive bias, like regularization that could induce a bias in predictions?

Kate Saenko: No, what I mean is even more broad and general: an algorithm that makes decisions in a biased way, biased against some attribute of the data. Let's say it's very accurate on daytime images, but very inaccurate on nighttime images. In the popular media we hear this more often referring to people and their demographics—for example an algorithm that's very accurate on light-skinned faces and very inaccurate on dark-skinned faces. That's the general biased algorithm definition: it's just not fair across different attributes of the data.

Sanjay Arora: Let's talk a bit about the social ramifications of this research. Did they play a part in your process of getting interested in bias in models and datasets?

Kate Saenko: Not in the beginning. I first became interested in the issue of dataset bias and domain adaptation when I was a PhD student. I was trying to train object recognition models and then put them on a mobile robot. The goal was to have a robot that you can tell, "Can you bring me a cup of coffee?" Or, "Can you at least recognize the cup of coffee?" I trained the algorithm on images that I got from Amazon.com. And then I tried using it on images that the robot captured in an indoor environment in the lab. It failed in a pretty dramatic way, even though when I tried it on the Amazon.com images, it worked with much higher accuracy. So that's what got me interested in this phenomenon: that just this shift of the domain from product images on Amazon to those in the real world destroyed the model's ability to generalize and do a good job.

Since then I have come across more social implications of dataset bias, and have studied some of them, in particular with regard to gender bias. We have a paper on that called "Women also Snowboard: Overcoming Bias in Captioning Models," where we look at how models can become biased across the gender dimension. Take for example models that do captioning for photos. They take an image and generate a caption of what's in the image. And if there are

people in those images, these models might say "Man" more frequently than "Woman," or some gendered word to describe the person. That certainly has more of a social implication. But I would say most of my research into dataset bias is not with data about people. It's more with objects and recognizing their differences.

So now the problem becomes, how do I train a classifier using the labeled source domain, which is the Amazon.com domain, and the target domain, which is the robot images that are unlabeled? How do I use those two datasets and train a model that is accurate on the target domain? That's what I've been working on, for the most part, over the last ten years. And we've made a lot of progress.

The only way that we can move forward in our research field is if the next researcher can take your results and build on them.

Sanjay Arora: Is there a certain fixed direction you have in mind where you want to take your research in the next five years or ten years, and are there specific sub-problems that interest you on that timescale?

Kate Saenko: I want to continue pushing in the direction of making domain adaptation work better. I'm also interested in expanding that definition and saying, well, what if we don't observe our target domain? How do we make sure the model is working well? What if the categories change? So, say in my training set I had 125 different objects, but I let my robot out into the world and it's starting to see objects that weren't part of the training set. How do we make sure that the robot doesn't try to classify them as something that is labeled in the source? For example, seeing a bottle of water and trying to classify that as a flower vase because it wasn't trained on water bottles, instead of just being able to say "Oh, that's a new category I don't know about."

Looking further or looking at a higher level, how do we even know that the model is looking at something that is out of its training domain? Actually, we don't have very good ways of knowing. So in some ways the minimum I would like my AI model to do is, if it's faced with data it hasn't seen or it's not able to adapt to or recognize, at least throw up its hands and say, "I know nothing about this," or "I don't feel comfortable with this domain." Humans would do this, right? If you train someone to be a very good classical piano player and then you say "Here's a very difficult jazz piece," they'll say "No, this is not what I do." The machine by contrast is just going to try to keep playing jazz at that point and do a bad job, because it doesn't realize you're giving it something that it wasn't trained on.

Sanjay Arora: This is a bit more of the technology and computing side, but what's your experience been with open source, and especially open source machine learning frameworks and software and technologies in general?

Kate Saenko: Everything we do in my group, we open source usually once the paper is accepted for publication. We make the code for it open source, and that's more or less the standard in my field or in computer vision and other AI subfields. We open source the code and often also the model itself, so the trained model is open sourced.

The only way that we can move forward in our research field is if the next researcher can take

your results and build on them. If we don't open source our code, then if you invent a better way to do object recognition, and I come along and want to improve on it, but I want to keep your improvement because it really pushes performance up, but I *can't* use your improvement, whatever I do is still going to be below that level. And so I would need your code to be able to build on your improvement, and if we don't open source your code, we just move much slower as a field.

Sanjay Arora: I'm guessing most of the frameworks that you use and the tools that you use are also open source, so PyTorch or the Python stack?

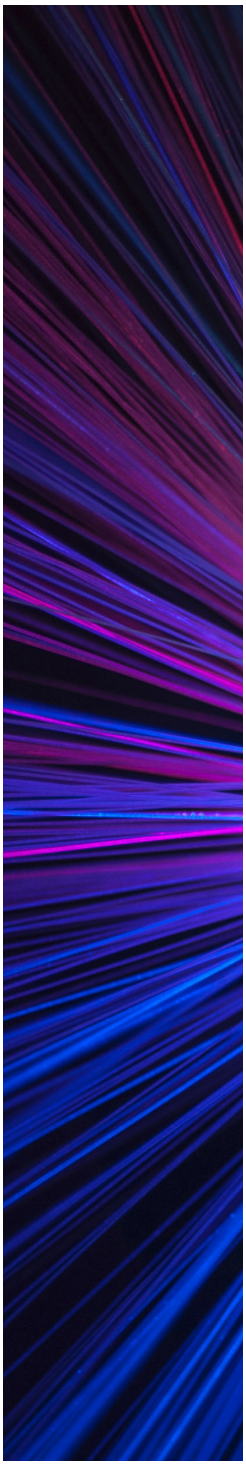
Kate Saenko: Yes. We use only open source tools in my research. In fact, in the beginning of this deep, large neural network revolution, the initial papers that published very good results with these large neural networks trained on a large amount of data, they weren't open source implementations. And even though it seemed exciting, people tended to feel like, "Well, we can't even verify or use it in any way." And so the lab I was in at Berkeley, I was a postdoc at the time, decided, "Well, let's just reimplement what this paper did and make it open source," and that became this library known as Caffe. That was a huge thing. Even a lot of companies started using Caffe because there was no other open source library. Later many open source versions of deep learning libraries came out, but for a while, Caffe was the main one.

Sanjay Arora: Does sharing of datasets play a role in AI research that might be different from other research you have done?

Kate Saenko: It's a little hard for me to tell because I'm not intimately familiar with how research is done in other areas of computer science. But data is the bloodline of the core of AI research. Just like we open source all of our algorithms, we also open source all of our datasets. If you try to submit a paper to a conference that is only evaluating on a proprhealietary dataset, that's not going to get past the review because you can't even reproduce somebody's research result if you don't have their data. You can have just their model, but if you want to run it or train it on your own dataset, you just have to be able to have the data and the model and the code.

We're getting better at that. It's still not perfect. People don't open source their code right away, or they do but it's buggy, or it's incomplete. My students run into these issues a lot, and I'm sure we're also guilty of it sometimes, but it's getting better.

On the other hand, I just thought of an example where this is not the case, and this is in the medical AI field: natural language processing of health-related patient documents, or computer vision analysis of medical scans. So in that field, because of Institutional Review Board (IRB) considerations and human subject considerations, data has been very difficult to share. So IRBs at institutions and hospitals will not let people share the data, for the most part. You can work on



it in house or you have to jump through a lot of hoops to get access to it. That has slowed down the progress in that field tremendously.

Sanjay Arora: I want to move on now to another interesting area you mentioned in your Red Hat Research Day talk. One of the papers you mentioned there is on building features that are invariant to distributional changes. They have a backbone with a classifier for the labels as well as a classifier that was predicting whether the data came from the source or target distribution. The idea was to train the label classifier to minimize the loss but use the source-target classifier to maximize the loss so that it's hard to distinguish between the source and target distributions and use those updates to generate invariant features in the backbone. Basically, a generative adversarial network (GAN).

Another relatively recent idea is building equivariance/covariance into the networks in the sense that the features in each layer transform under a certain group of symmetries. For 3D rotations, this would be SO(3), for example. This is a vague question, but are there approaches that try to learn the group of symmetries that map from one domain to another and build these equivariances into the network.

Kate Saenko: So convolutional neural networks are already somewhat invariant to translation. If you move an object in the image, they are invariant to that. I think there was a lot of work in building invariances into these networks, but I think the visual world is very, very diverse, and at some point we just don't know what we need the invariance to.

One thing that people do a lot is normalization, so normalizing your data, normalizing the outputs of layers to make sure that everything is at least on the same scale so you don't have huge values coming in and they somehow throw everything off. But I would say, more recently, as a field we are starting to become interested in learning without any labels—unsupervised learning. For that, we're using some of these invariances that we know of. So we know that if I'm looking at an object and I rotate that object slightly, it doesn't change the identity of the object. Or if I add some small amount of noise to the image, a dog still looks like a dog. Or even if I randomly change the color, it might become a pink dog, but it still looks like a dog and you're still going to classify it as a dog.

So we know some of these invariances that apply to our visual tasks that we're studying, and we're using them to train models to become invariant. But the way we're doing it is not by changing the structure of the network, but rather, by producing them as data augmentations. We take the training image of the dog and we augment that training image by producing versions of it that have these additional variations in them, like slight rotation, cropping, adding noise, changing colors, contrast, and so on. Then we give that to the network and say, "This is still the same object. So learn that this variation we just added to it shouldn't matter, as far as predicting that object."

Sanjay Arora: That makes sense. At least a couple of papers I looked at were using GANs or versions of GANs to map between the source and the target distributions and back, mapping the distributions themselves. Is that the most common way of trying to fix this distributional mismatch or are there other techniques too?

Kate Saenko: I think there is some work that does that, but you have to realize that GANs are also quite brittle. To get a GAN to generate a realistic-looking face in a video, people have put lots of work into that. Now we have StyleGAN and StyleGAN2 generating really realistic-looking

pictures of people who do not exist. But for pretty much any other type of visual object, they still don't work that well.

So even though people have tried using GANs to translate between domains, it's still very nascent, not at the point where it just works on any domain you want. I mean, I think for some very small changes in the domain, you could use a GAN, like if all you did was take your dataset and try to use the same dataset but with fog added to it or with snow added. I think a GAN can learn to generate that. But if you now said, this is an extreme viewpoint change, and you want the GAN to generate complete scenes of traffic scenes, but from a different viewpoint, it's going to really struggle with that.

Sanjay Arora: Is there any work on generalizing domain adaptation or applying it to reinforcement learning to adapt across environments that are a bit different? A typical example: you have a simulated environment where you're teaching a skeleton to run, and let's say now you wanted to run on a mountain. So of course your running policy changes a bit, you're bent forward a bit and things like that. Is there any work being done on applying domain adaptation techniques there in reinforcement learning?

Kate Saenko: I think that there are definitely some techniques being applied, especially when you're dealing with visual inputs. You're learning from visual input, as opposed to learning in a simulated environment. We have some work on that: we train a robot arm to pick up objects and manipulate objects. But because robots break easily or they take a long time to do the trial and error that you need to train a policy, we train them in simulation.

Ninety-nine percent of the reinforcement learning papers I see do everything in simulation, but simulation is a very narrow domain. You're controlling everything.

However, there's a huge domain gap with robotics. If you train your biped robot to run on a domain, and then you put that same algorithm on a real robot, what is it going to do? Is it going to run? It's not going to work. Ninety-nine percent of the reinforcement learning papers I see do everything in simulation, but simulation is a very narrow domain. You're controlling everything.

And so broadly speaking, there's a huge domain shift in all of these reinforcement learning applications, but I don't think a lot of people are looking at that because not a lot of people are transferring their reinforcement learning techniques into the real world. There's work from UC Berkeley from a while back, and also from Google on this kind of object manipulation. We also looked at it with my colleagues at Northeastern. If you want to train a robot arm to manipulate objects and actually have that algorithm generalized to real world observations, you have to solve this problem and then you can use some similar techniques. We even used GANs to try to solve that problem.

Sanjay Arora: For some of our problems, a lot of them at the operating system level or the compiler level, you actually just run the compiler, you run the process in your OS. Of course there are many hardware simulators, for chips or memory accesses for example, and you're trying to learn a policy, but it's a simulation. And then, like you said, once you transfer it, it does horribly. So just trying to minimize that domain mismatch is a pretty hard problem.

Kate Saenko: I don't know what the right techniques are there. My students have been looking at drone control and trained a neural network policy to control a small drone, just flight control. And it trained very well in simulation, it achieved high reward. It followed the directions given to it, to control the robot. They put it on the drone, and the thing crashed and almost set the place on fire. And they spent at least probably six months to even closer to a year trying to fix and figure that out, and we have a paper now that fixes it, but it's a problem.

Sanjay Arora: And especially when something actually crashes and burns, I mean, that psychologically hurts too.

Kate Saenko: Yes, it literally crashed and burned.

Red Hat

RESEARCH QUARTERLY

VOLUME 2:4

Feature

Sequential Monte Carlo for streaming data

Bayesian statistical methods can make predictive data analysis more accurate. In this article, we evaluate possible solutions to the challenge of refining and increasing the value of high-volume data streams.

About the Author

Rui Vieira is a PhD graduate from the School of Mathematics, Statistics, and Physics at Newcastle University, UK. During his PhD, his research focused on statistical inference for long-running time series, particularly on Sequential Monte Carlo methods applied to Dynamic Generalized Linear Models. At Red Hat, he has worked with distributed computing applied to intelligent apps, and his current focus is on the integration between machine learning and business automation.

by Rui Vieira

When dealing with an overwhelming amount of data, it is often beneficial to process the data locally as it is collected, at the device level. Take for example the common case of IoT devices continuously providing sensor data: current location and temperature, accelerometer readings, light and humidity levels. All this information is likely to be sent at high frequency and be very "noisy." Data like this could be consumed by user-facing applications or used to train machine learning/AI algorithms to provide additional insights, but those uses will be much more effective if the data is cleaned up and enriched before processing.

What kind of "enrichment" is useful? We might want to know if the observation is anomalous, i.e., outside a quantifiable threshold. We might want to use de-noised data. We might want to use a short-term prediction for a time series, or we might want a method to interpolate missing points. Bayesian methods provide tools that can answer these questions in many scenarios. We will look at some of these available methods, but first, we'll define some of our terms.

Real time refers to methods that can provide a result within a fixed, bound computational time and that are compatible with a human perception of real time, e.g., in the sub-second scale. Real-time estimation is also a necessity when dealing with inference in the context of potentially mission-critical systems, such as self-driving cars, drones, or any other system that requires an "always on" state, usually with a high-frequency input of data.

Online means an inference method that takes one observation, or at least a fixed window of observations, into account. That is, we want our inference method to be *O(1)* relative to the number of observations, without growing in computational cost as time goes by (e.g., as a result of having to process historical data). As a practical example of an online method, we can consider filtering location and position data from a car's GPS. We wouldn't expect the computational cost to grow unbounded as we drove for longer!

Having a strong statistical underlying foundation is also essential if we want to be able to make critical

decisions based on error margins or confidence of our estimations. For instance, if a sensor is missing a few observations, we can carry forward the state and quantify the prediction's confidence interval. Also, we might want to classify an observation as an outlier if it falls outside a quantifiable threshold. These tasks are simple to perform if we choose the appropriate model for our data.

PARTICLE FILTERS AS A POSSIBLE SOLUTION

A standard way of performing estimation on time series and guaranteeing the online property is to use state-space models (SSMs). In state-space models, we assume that an underlying hidden state evolves in time and that at each time point we measure an observation. SSMs possess a *Markovian nature*; that is, at each time point, the current state only depends on the previous state and model's parameters, and observations rely on the current state and model's parameters.

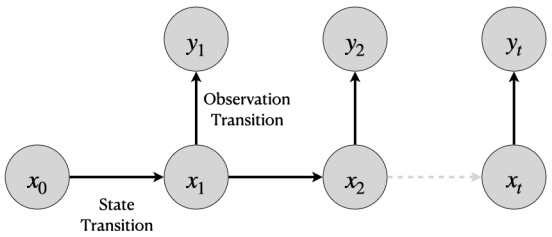


Figure 1. *State-space model showing hidden states* x*, and corresponding state transition and observations* y*, with observation transition*

For a specific class of models where we assume linearity, i.e., an underlying Gaussian posterior for both state and observation transitions, some classic and exceptionally well-researched solutions are available, e.g., the venerable Kalman Filter (KF) introduced by Kalman et al. in 1960. The KF provides the optimal solution for this model family. But even in the case where we don't have a linear system, there are other variants of the KF, such as the Extended Kalman Filter (EKF). Proposed by both Jazwinki (1966) and Maybeck (1979), the EKF can be used by approximating the non-linear transitions with a first-order Taylor Series. However, different approaches are available that neither assume state model linearity or unimodal posteriors, nor rely on local linearization.

One such solution is called Particle Filters (PF), also commonly known as Sequential Monte Carlo (SMC), a well-established method of Bayesian inference based on *importance sampling*. These methods have been well researched since the latter part of the twentieth century. They have been increasingly adopted in a multitude of scenarios in recent years, made possible with the advent of increased computational power.

In a nutshell, PFs estimate a sequence of underlying states, as defined by our SSM. This is done by propagating a certain number of particles according to a state transition. Numerous algorithms are available, but in a standard PF (e.g., Sequential Importance Resampling [SIR]), we resample particles according to their weights, provided by the observation's likelihood, and the resulting particles are propagated further.

The underlying concept is to try to explore the state-space as efficiently as possible, thrusting particles forward and creating an approximation of the state's posterior. Although theoretically the PF estimation error converges to zero as the number of particles increases to infinity, in the real world we are dealing with a finite (and sometimes extremely constrained) number of particles. The strategy therefore is to select the particles with a higher likelihood, according to our observation model, and duplicate them. With a finite number of particles and without resampling, most of the weights would eventually be zero.

In a nutshell, PFs estimate a sequence of underlying states, as defined by our SSM.

So how is the state estimation determined from this set of particles? Depending on the implementation used, we can choose a weighted mean of the particles or a single particle with the highest weight.

THE PROBLEM OF PARTICLE IMPOVERISHMENT

Although PFs provide the framework for online state estimation in SSMs, we need to define our transition and observation models. PFs coupled with a flexible, elegant way of defining our underlying transition and observation model, such as the Dynamic Generalised Linear Models (DGLMs), provide a compelling solution.

DGLMs allow us to model complex time-series behaviours by composing simple patterns into more complex ones. We have, for instance, components for linear and polynomial trends, as well as cyclic trends represented by Fourier components. We assume a linear state transition, but can now associate it with a linear or nonlinear observation model. This gives us the ability to model a wide variety of data—e.g., both continuous and discrete data with complex cyclic patterns.

Special consideration must be made, however, when using PFs for low-powered hardware, such as that typically found in IoT devices. An obvious issue is that computational requirements grow linearly with the number of particles and also grow with the state-space dimensions. A tradeoff must be made between the computational cost and the estimation error, as well as the model's complexity. As an example, is it necessary to incorporate a yearly seasonal component in a temperature data stream model, if the time-series granularity is the order of a measurement every second?

In any case, the act of resampling the particles introduces a problem usually termed *particle impoverishment*. If we think of selecting a portion of particles with higher likelihood and discarding the remaining ones, we are throwing away information. After a few iterations, all the particles will have a single ancestor. This is problematic since we now are not exploring the state-space as efficiently as possible.

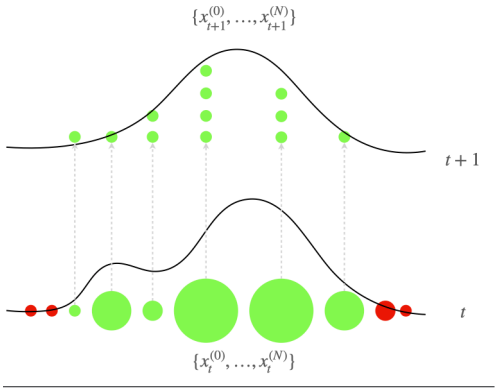


Figure 2. *Illustration of resampling and particle impoverishment*

Clearly, these two problems—computational costs and reducing particle impoverishment (especially important for long-running time-series data as we might find in IoT devices)— constitute two of the biggest challenges when implementing local stream processing in low-powered devices. We will look at some of the potential solutions.

With modern high-end hardware, we could simply use a massive number of particles. This might not necessarily introduce a computational cost problem, since PFs are notoriously, embarrassingly parallel in the state propagation phase (i.e., each particle can be propagated independently, in parallel), and this property could be further exploited with modern hardware architectures, such as high numbers of CPUs, as well as GPUs.

This is, however, problematic in low-powered devices. Additionally, increasing the number of particles will only delay the problem; we are just postponing the inevitable particle impoverishment, and the computational costs will be prohibitive. Solutions will have to rely on theoretical advancements, rather than just brute force calculations.

Another consideration is that, so far, we have assumed that for "standard" PFs, we are only interested in estimating the state. But that implies that we are assuming that we know the state and observation's transition model parameters. This is often not the case.

The estimation becomes even more complex when we are trying to estimate the model's states and parameters simultaneously. And this may very likely be the real-world scenario we are dealing with.

HOW SMC2 COULD HELP

Over the years, a considerable amount

of research has been made into the state and parameter estimation. A promising solution is to use methods such as SMC^2, as introduced by Chopin et al. (2013). SMC^2 provides the ability to estimate both states and parameters. However, this requires the ability to calculate the observation's incremental likelihood at each time point, which, for most models, won't be possible in an analytical way. Crucially, Standard PFs (e.g., SIR) provide an unbiased estimator of the marginal likelihood and can be used as "parameter" particles. This nested PF-within-a-PF approach is alluded to in the name SMC-squared.

Since, as mentioned earlier, SIR filters rely on the assumption that the parameters are known, we condition each of these Parameter PFs on a particular parameter value. A degeneracy threshold is established, and if that threshold is crossed, then the parameter particles are resampled. The resampled particles can then be propagated using a Particle Markov chain Monte Carlo method (PMCMC) step, such as Particle Marginal Metropolis-Hastings (PMMH) or Particle Gibbs. This step is essential to the algorithm because it permits performing a particle "rejuvenation" by using a PMCMC kernel, which can greatly help to mitigate the problems of particle impoverishment. However, because this rejuvenation step uses previous data, it ceases being an online method (although it is still a sequential method), requiring the storage of past stream data points.

However, it is possible to keep the computational cost bound per iteration, by targeting an approximate distribution and evaluating the PMCMC for a fixed window of recent observations. Ad-hoc implementations, such as O-SMC^2, allow use of the flexible and model-agnostic advantages of SMC^2 while keeping the sequential and online requirements for streaming data.

> The estimation becomes even more complex when we are trying to estimate the model's states and parameters simultaneously.

Another potential solution to be considered is the state augmentation family of PFs. One example is Particle Learning (PL), as presented by Carvalho et al. (2010). This approach uses an *essential vector*, a vector of sufficient statistics that summarises the model state used for the state and parameter propagation itself. If each particle has an associated set of sufficient statistics, these can be individually updated, at each time-step, in a deterministic way. This has the obvious advantage of providing a low computational cost due to the low dimensionality of the sufficient statistic vector. Additionally, provided sufficient statistics are available, this marginalisation of the state and parameters can help fight particle degeneracy. In benchmarks, PL methods provided a reasonable approximation to the true state (estimated offline, using a gold standard such as PMMH).

Methods such as O-SMC^2 provide attractive features, such as the ability to apply to a huge variety of models, lower estimation error, particle rejuvenation, and being implementation agnostic. However, the computational cost might be too high for low-powered devices dealing with high-frequency data, in the magnitude of at least a few seconds for each time point on modest hardware. For streaming data with lower frequency (e.g., data points every minute), it might be a compelling option. Sufficient-statistics based methods, such as PL, provide a reasonable real-time estimate, and in theory should delay particle impoverishment issues, at the cost of a few hundred milliseconds per iteration. This is especially the case when our models are simple (i.e., few state components), for instance, for temperature estimation or motion estimation.

In summary, although theoretical research is extremely active in the field of real-time, online state and parameter estimation for streaming data, if the goal is to provide the best possible estimates, PFs are still not a match for offline methods such as the Markov chain Monte Carlo method. However, if the goal is to have a reasonable estimation sufficient for IoT data processing, PFs offer a compelling, flexible, and robust solution, with methods that provide strong candidates for further research in this field.

REFERENCES

Andrieu, C., Doucet, A. and Holenstein, R. (2010). "Particle Markov

chain Monte Carlo methods." *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 72 (3), 269–342.

Carvalho, C.M., Johannes, M.S., Lopes, H.F. and Polson, N.G. (2010). "Particle learning and smoothing." *Statistical Science* 25 (1), 88–106.

Chopin, N., Jacob, P. E. and Papaspiliopoulos, O. (2013). "SMC^2: An efficient algorithm for sequential analysis of state-space models." *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 75 (3), 397–426.

Jazwinski, A. (1966). "Filtering for nonlinear dynamical systems." *IEEE Transactions on Automatic Control* 11 (4), 765–66.

Kalman, R. (1960). "A new approach to linear filtering and prediction problems." *Journal of Basic Engineering* 82, 35–45

Maybeck, P. S. (1979). *Stochastic models, estimation, and control, vol. 1.*

Vieira, R. (2018). "Bayesian online state and parameter estimation for streaming data." PhD diss., Newcastle University. http://theses.ncl.ac.uk/jspui/handle/10443/4344

Vieira, R., and Wilkinson, D. (2016). "Online state and parameter estimation in dynamic generalised linear models." arXiv preprint arXiv:1608.08666.

RESEARCH
QUARTERLY

VOLUME 2:4

Red Hat

Feature

Blocks, microworlds, puzzles, and adaptivity: teaching programming effectively

Programming can be a difficult skill to learn, especially for younger students. Understanding how learning works and applying that knowledge to creating engaging exercises can make a big difference when teaching novices.

By Tomáš Effenberger

Why teach programming? Efficient and engaging learning helps children to become competent and autonomous. Being competent and autonomous, they are able to improve everything else. Problem solving, in particular, is among the universal competencies needed for contributing to society. Introductory programming is an expedient modern approach to teaching problem solving, as it fosters key skills like pattern recognition, abstraction, and problem decomposition, which are collectively labeled as computational thinking.

There is an abundance of strategies that have been shown to increase learning gains. For example, interleaving topics increases the mental effort required to solve problems, and increased student activity leads to better learning. But no strategy is universal. If the current topic is already too difficult for the student, making it even more difficult is likely to further impede learning. To understand why and when specific strategies work, it is therefore helpful to first take a closer look at learning itself.

PRINCIPLES OF EFFECTIVE LEARNING

Instructional strategies that fulfill innate psychological needs increase intrinsic motivation, which leads to greater attention, effort, and, eventually, learning. Although needs differ among students, *self-determination theory* identifies three universal needs that frequently drive our decisions: competence, autonomy, and relatedness.

In addition to basic motivation, there is another key ingredient for learning: a suitable learning activity. If the activity is too easy, and the student does not need to exert effort, little learning will happen. Similarly, little learning will happen if the activity is too difficult for the student. For efficient learning, the activities should be within the reach of the student—possibly with some support, yet not fully automated. This sweet spot of appropriate difficulty is called the *zone of proximal development*.

A mismatched difficulty level also leads to negative emotions: boredom if the activity is too easy, frustration if it is too difficult. These negative emotions hurt motivation, consequently decreasing attention, leading to even lower learning. On the other hand, appropriate, skill-stretching challenges support

About the Author
Tomáš Effenberger is a PhD candidate in Computer Science at Masaryk University, where he also received his master's degree. After research internships at the University of Oxford and Google AI, he became a member of the Adaptive Learning group at Masaryk University. His mission is to make learning more efficient and engaging by advancing techniques for improving adaptive learning systems.

achieving the *state of flow*, in which the student is completely immersed in the learning activity. According to the flow theory, there are two additional properties of activities needed to achieve the state of flow: immediate feedback and a clear goal.

Finally, learning activities should account for the severe limits of working memory. *Cognitive load theory* divides cognitive load into three types: extraneous, intrinsics, and germane. Extraneous cognitive load is the effort not directly related to the topic, intrinsic cognitive load is the effort associated with the inherent complexity of the topic, and germane cognitive load is the effort used for actual learning. For efficient learning, cognitive load theory recommends minimizing the extraneous load, controlling the intrinsic load, and maximizing the germane load. The high intrinsic complexity of programming explains why it is so difficult to learn: students need to learn the syntax of a programming language, its semantics and pragmatics (when to use which programming construct), as well as problem-solving and debugging strategies—all at the same time.

STRATEGIES SUPPORTING LEARNING

To decrease the otherwise excessive intrinsic cognitive load, introductory programming exercises can feature a block-based programming interface to avoid syntax errors, visualization of program execution in a microworld, and scaffolding in the form of a starter code. The intrinsic load can be further decreased by *pretraining*, or preceding programming puzzles by worked examples, visualizations, and problems on understanding program execution.

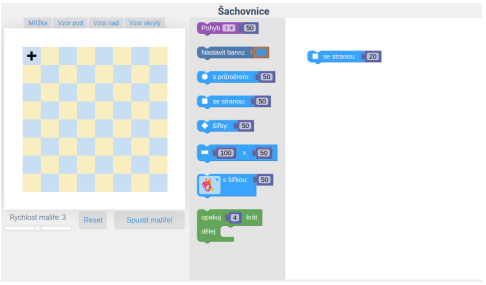


Figure 1. *A block-based programming interface in Umime Programovat*

By no means, however, should the learning be effortless. As suggested by the zone of proximal development, too-easy problems are not helpful. We should strive to maximize the germane cognitive load. Active problem solving, for instance, usually leads to better learning outcomes than passively watching a lecture. There are other strategies to achieve desirable difficulties. For example, a cognitive conflict — revealing a student's misconception — increases attention and helps to build more viable mental models of programming concepts. Interleaving of topics allows practice in not only *how* to use programming constructs, but also *when* to use them. Personalized problem recommendations can be employed to select appropriate difficulty based on the current skills of the student.

Other strategies focus primarily on motivation. Many learning systems provide mostly extrinsic motivational elements like points and badges. While useful to increase short-term performance, the extrinsic motivation can sometimes lead to detrimental effects, such as engaging in inefficient but rewarded activities and even undermining the original intrinsic motivation. Strategies that support intrinsic motivation, by fulfillment of psychological needs, should be the primary focus.

The theory of flow suggests three strategies for intrinsically motivating activities: difficulty that matches the skill of the student, well-structured problems with a clear goal, and immediate formative feedback. These strategies are also consistent with self-determination theory, as they support efficient progress towards mastery, which is closely related to the need for competence. Competence can be further promoted by progress visualization and praise. In order to increase long-term intrinsic motivation, praise should be sincere, specific, focused on the process rather than results, and relevant to attainable but nontrivial standards.

The need for autonomy can be fulfilled by giving students some level of control over their own learning. However, since students are not typically able to manage their own learning very well, it is still useful to provide at least soft recommendations, which the students are not forced to follow.

Finally, intrinsic motivation can be further increased by an entertaining story and

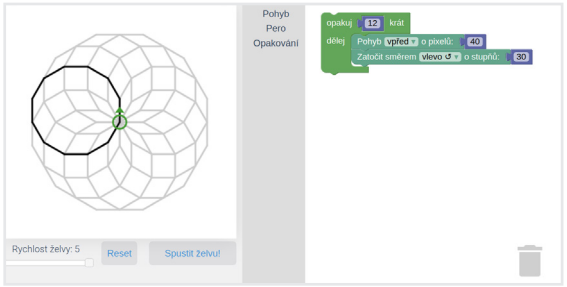
appealing microworld, which can fulfil the innate needs for playfulness, curiosity, and harmony. Ideally, the story should be *endogenous*, or directly related to what is learned. A story that is not related to the learning content, such as collecting points for correctly solved math problems, is sometimes called a "chocolate-covered broccoli", as it only provides extrinsic motivation.

INTRODUCTORY PROGRAMMING EXERCISES

Many of the discussed strategies can be applied directly to programming exercises. A popular example is the use of block-based programming interfaces. Block-based programming decreases intrinsic cognitive load, since the students do not need to remember available commands and their syntax. Furthermore, it makes the structure of the code more visible and helps to build more viable mental models by providing an expert-level view of the code structure.

Another typical feature of introductory programming exercises is a microworld. Visualization of program execution in a microworld decreases intrinsic cognitive load and helps to build a correct mental model of execution, while simultaneously supporting intrinsic motivation by fulfilling the need for competence (when allowing for powerful effects) and playfulness. Typical examples of microworlds are a robot on a grid and turtle graphics.[1]

Figure 2. *Turtle graphics in Umime Programovat*



> The theory of flow suggests three strategies for intrinsically motivating activities: difficulty that matches the skill of the student, well-structured problems with a clear goal, and immediate formative feedback.

Open-ended environments with a block-based programming interface and an appealing microworld can promote autonomy and curiosity, but the lack of guided instruction can result in adoption of bad programming habits. As an alternative to the open-ended exploration, a learning system can provide a series of programming puzzles. Short puzzles – if well designed and ordered from the easiest to the most difficult – can greatly support the state of flow. They provide clear attainable goals, immediate feedback on the performance, and also an appropriate level of challenge. The need for autonomy can be satisfied by other means, for example, by letting the student choose from multiple exercises available in the learning system.

Programming exercises can draw inspiration from successful puzzle games, which share common design principles, such as completing incomplete patterns and gradually unfolding a single path to solution through a sequence of small steps. Various forms of scaffolding, such as an initial partial program to complete, can be used to allow the students to solve interesting challenges from the very beginning.

The individual puzzles are important, but their context in the learning system can greatly impact the experience as well. This

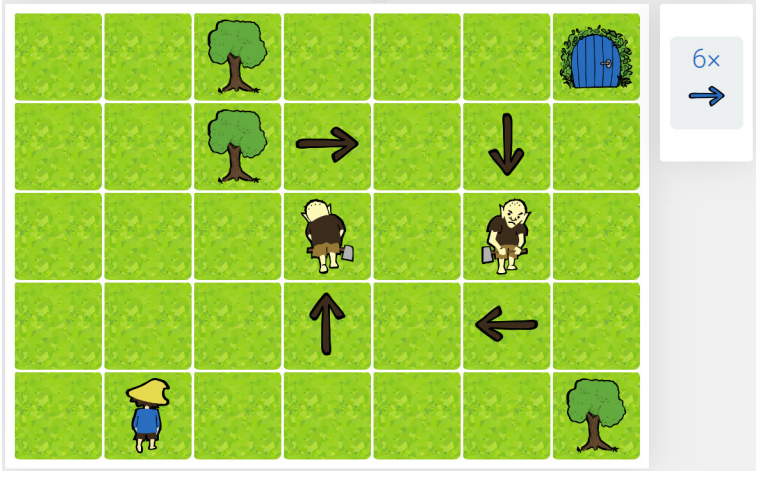
1. You can see various microworlds used in programming exercises at https://www.umimeprogramovat.cz and https://en.robomise.cz.

includes ordering of problems, tracking and visualization of progress, and recommendations on which problem to tackle next. To support the appropriate level of challenge, the puzzles should be ordered from the easiest to the most difficult. The difficulty, however, is not the only aspect for a successful puzzle progression. For instance, novelty in game elements is important to keep the puzzles engaging.

ADAPTIVE LEARNING

Adaptive learning refers to the use of data science to support learning and motivation, for example, by selecting problems of the optimal difficulty, optimizing parameters of mastery learning, and iteratively improving the educational content. Adaptive learning builds on the principles of effective learning and leverages methods from statistics, machine learning, and recommender systems. Rather than aiming at complete automation using artificial intelligence, however, the best results are usually obtained by appropriately combining human and machine intelligence. Such an approach is called *intelligence amplification*.

Figure 3. *Arrows exercise in Umime Programovat*



Iterative development of programming exercises is a good example of such human-machine synergy. Humans are still much better at creating engaging puzzles, but they struggle to estimate how difficult the puzzle will be for the novice programmer. Analysis of collected data can reveal that some problems are considerably easier or more difficult than originally expected or that the students solve the problems without the intended programming construct (e.g., without loops).

Such insights can help us to iteratively improve the exercises by reordering the puzzles, clarifying the problem statements, or providing hints, scaffolding, and feedback. Analysis of common misconceptions can sometimes also lead to ideas for new puzzles to create. This agile and research-based approach proved useful for developing content for https://www.umimeprogramovat.cz, a learning system that contains diverse programming exercises, some suitable for primary school children, others targeting high school and university students.

For more detailed discussion of the guidelines for design and iterative improvement of block-based programming puzzles, read "Design and analysis of microworlds and puzzles for block-based programming," recently published in *Computer Science Education*.[2] RH RQ

Acknowledgements

Thanks are due to Radek Pelánek for supervision and Red Hatters Marek Grác and Milan Brož for facilitation of the academic cooperation of Red Hat Czech and the Faculty of Informatics at Masaryk University.

2. The paper is available at https://doi.org/10.1080/08993408.2020.1832813, or as a preprint at https://www.fi.muni.cz/adaptivelearning/?a=publications.

RESEARCH
QUARTERLY

VOLUME 2:4

Red Hat

Feature

Efficient runtime verification for the Linux kernel

If safety-critical systems fail, they can cause significant damage, including loss of life. In this article we consider methods to verify their behavior in production.

by Daniel Bristot de Oliveira

This is the second of a series of three articles about the formal analysis and verification of the real-time Linux kernel. Read the first article in RHRQ *2.3.*

The challenge to enabling Linux in safety-critical applications is the requirement for a new set of tools that can demonstrate evidence that the system can achieve some level of safety. For instance, for time-sensitive systems we would want to see a collection of evidence demonstrating that real-time capabilities need to be provided. Such a level of evidence depends on the criticality of the system, and it can range from continuous testing and documentation to the application of formal methods, the latter being the most sophisticated approach.

Formal methods consist of a collection of mathematical techniques to rigorously state the specifications of a system. The advantage of using mathematical notation is that it removes the ambiguous nature of natural language. The mathematical notation also enables the automatic verification of the system. Despite the arguments in favor of formal methods, its application is generally restricted to specific sectors. The most commonly mentioned reasons for that are the complexity of the mathematical notation used in the specifications and the limitations of computational space and processing time required to verify a system using formal methods.

In our previous article, we presented a formal specification approach using automata-based models, showing that it was possible to build a formal model of thread synchronization for the real-time Linux kernel. The final model accounted for more than 9,000 states and 21,000 transitions. However, it was built from reasonably small specifications, which made it practical for modeling purposes. We then need to ask: How to take benefits from the model? How can we check if the system adheres to the model?

Moreover, for safety-critical applications, it is as important to demonstrate the properties of the system as it is to provide ways for the system to react gracefully to a failure. For example, we can make the system fall back to a fail-safe mode in the case of detected misbehavior. Hence, to be effective, the usage

About the Author
Daniel Bristot de Oliveira is a principal software engineer at Red Hat working in the development of real-time features of the Linux kernel. Daniel has a joint PhD degree in Automation and Systems Engineering at UFSC (BRA) and in Embedded Systems at the Scuola Superiore Sant'Anna (ITA). He is also a post-PhD researcher in the Retis Lab at the Scuola Superiore Sant'Anna.

The verification approach has three major phases: the modeling phase, the model-code transformation phase, and the runtime verification phase.

of the formal methods must also allow the runtime verification of the system.

In this article, we will summarize the results presented in the paper "Efficient formal verification for the Linux kernel." The paper proposed an efficient automata-based verification method for the Linux kernel, capable of verifying the correct sequences of in-kernel events at runtime. The method auto-generates C code from models, with efficient transition look-up time in O(1) for each hit event. The generated code can be then loaded on the fly into the kernel and dynamically associated with kernel-tracing events. This approach enables efficient runtime verification of the observed in-kernel events, as demonstrated in the next sections.

THE EFFICIENT VERIFICATION APPROACH

The verification approach is presented in **Figure 1**:

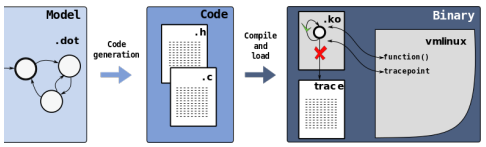


Figure 1. *Runtime verification approach*

The verification approach has three major phases: the modeling phase, the model-code transformation phase, and the runtime verification phase.

The modeling phase

First, an automata-based model of Linux's behavior is developed using the set of events available in the tracing infrastructure of Linux. (This modeling approach was presented in the first article of this series.) The model is represented using the Graphviz.dot open format. This format is widely used for automata models and can be exported from modeling tools such as Supremica.

The model-code generation phase

The automata formalism is better known by its graphical representation, as shown in **Figure 2**:

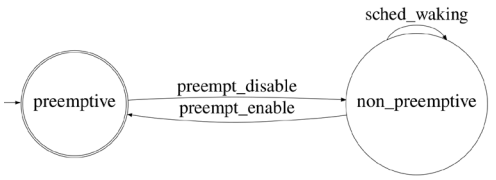


Figure 2. *Wakeup in preemptive model*

However, as the name suggests, the graphical format is just a representation of an automaton. A deterministic automaton, denoted by G, is actually formally defined by a tuple $\mathbf{G = \{X, E, f, x_0, X_m\}}$, where

– $\boldsymbol{X}$ is the set of states;
– $\boldsymbol{E}$ is the finite set of events;
– $\boldsymbol{f}$: $\boldsymbol{X} \times \boldsymbol{E} \rightarrow \boldsymbol{X}$ is the transition function that defines the state transition in the occurrence of an event from $\boldsymbol{E}$ in the state $\boldsymbol{X}$;
– $\mathbf{x_0}$ is the initial state;
– $\mathbf{X_m} \subseteq \mathbf{X}$ is the set of marked states.

The automaton works as follows. It starts at the initial state $\mathbf{x_0}$, and upon the occurrence of an event $\mathbf{e} \subseteq \mathbf{E}$ with $\mathbf{f(x_0, e)}$ defined, the state transition from $\mathbf{x_0}$ to $\mathbf{f(x_0, e)}$ will take place.

This process continues based on the transitions for which ***f*** is defined.

Exploring the formal representation, we developed a tool named *dot2c*. This *dot2c* translates the automaton in the .dot file into a C data structure. The auto-generated code follows a naming convention that allows it to be linked with a kernel module skeleton that is already able to refer to the generated data structures, performing the verification of occurring events.

```
enum states {
        preemptive = 0,
        non_preemptive,
        state_max
};

enum events {
        preempt_disable = 0,
        preempt_enable,
        sched_waking,
        event_max
};

struct automaton {
        char *state_names[state_max];
        char *event_names[event_max];
        char function[state_max][event_max];
        char initial_state;
        char final_states[state_max];
};

struct automaton aut = {
        .event_names = { "preempt_disable", "preempt_enable",
                         "sched_waking" },
        .state_names = { "preemptive", "non_preemptive" },
        .function = {
                        { non_preemptive,         -1,             -1 },
                        {             -1, preemptive, non_preemptive },
                    },
        .initial_state = preemptive,
        .final_states = { 1, 0 }
};
```

Figure 3. *The automaton from Figure 2 translated into C code*

```
char get_next_state(struct automaton *aut, enum states curr_state,
                    enum events event) {
        return aut->function[curr_state][event];
}
```

Figure 4. *Transition function in C*

Regarding scalability, although the matrix is not the most efficient solution with respect to the memory footprint, in practice, the values are reasonable for nowadays common computing platforms. For instance, the Linux Task Model Automata presented in the previous article, with 9,017 states and 20,103 transitions, resulted in a binary object of less than 800KB, a reasonable value even for current Linux-based embedded systems.

The benefits of the format come in the runtime complexity/overhead. For example, the transition function f in C, shown in **Figure 4**, returns the next state in constant time O(1). The same complexity repeats for the other automaton operations in C.

The runtime verification phase

With the code generated from the model, the next challenge was to connect the verification code with the respective kernel events. Linux has an advanced set of tracing features, which are accessed mainly via **perf** and **ftrace**. Both other tools can hook to the in-kernel trace methods, processing the events in many different ways. The most common action is recording events into a trace buffer for post-processing or human interpretation of the events. However, it is also possible to hook other functions to trace events. For example, the live patching feature of Linux uses the function tracer to hook and deviate a problematic function to a revised version of the function that fixes a problem.

Exploring the possibility of hooking functions to the trace events, we proposed hooking the functions that run the automaton to the kernel events, enabling the online synchronous verification of the kernel. In this way, anytime a kernel event is generated, the function that parses the automaton is called, verifying if the model accepts the occurrence of that event. If the event is expected, it can be either logged or ignored. Otherwise, actions can be taken, ranging from the log of the exception to the halt of the system, to avoid the propagation of a failure.

Hence, the verification mechanism has a two-fold contribution to the application of Linux on safety-critical systems: it enables the continuous verification of the kernel against formal specifications, and it allows prompt reactions to the occurrence of unexpected events.

However, to be practical, the overhead of the verification needs to be acceptable for production systems. This aspect is explored in the next section.

PERFORMANCE ANALYSIS

We demonstrated the performance

of the proposed technique by presenting evaluation results on a real platform, verifying models in terms of the two most important performance metrics for Linux developers: throughput and latency.

> ...we present an approach that makes the runtime verification of the Linux kernel feasible.

The measurements were conducted on an HP ProLiant BL460c G7 server, with two six-core Intel Xeon L5640 processors and 12GB of RAM, running a Fedora 30 Linux distribution. The kernel selected for the experiments is the Linux PREEMPT_RT version 5.0.7-rt5.

Throughput evaluation was made using the Phoronix Test Suite benchmark, and its output is shown in **Figure 5**. The same experiments were repeated in three different configurations:

1. The benchmark was run in the system as-is, without any tracing or verification running.
2. It was run in the system after enabling verification of the SWA model.
3. A run was made with the system being traced, only limited to the events used in the verified automaton.

It is worth mentioning that tracing in the experiments means only recording the events. The complete verification in user space would still require the copy of data to user space and the verification itself, which would add further overhead.

On the CPU bound tests (Crypto, CPU Stress, and Memory Copying), both trace and verification have a low impact on the system performance. In contrast, the benchmarks that run mostly on kernel code highlight the overheads of both methods. In all cases, the verification performs better than tracing. Despite the efficiency of tracing, the amount of data that has to be manipulated costs more than the simple operations required to do the verification, essentially the cost of looking up the next state in memory in O(1), and storing the next state with a single memory write operation.

Latency is the main metric used when working with the PREEMPT_RT kernel. The latency of interest is defined as the delay that the highest real-time priority thread suffers during a new activation due to in-kernel synchronization. Linux practitioners use the cyclictest tool to measure this latency and rteval as background workload, generating intensive kernel activation. Like the previous experiment, the verification of a model (the NRS model in this case) was evaluated against the kernel as-is, with the kernel merely tracing the same set of events. Consistent with the results obtained in the throughput experiments, the proposed verification mechanism is more efficient than the sole tracing of the same events while keeping the overhead low.

FINAL REMARKS

In the previous article, we showed that it is possible to model Linux's complex behavior using automata-based specifications. However, that work would not be practical without a way to verify that the kernel and the model adhere to each other. In this article, we present an approach that makes the runtime verification of the Linux kernel feasible. We achieved this result by reducing the complexity of transforming a model into code

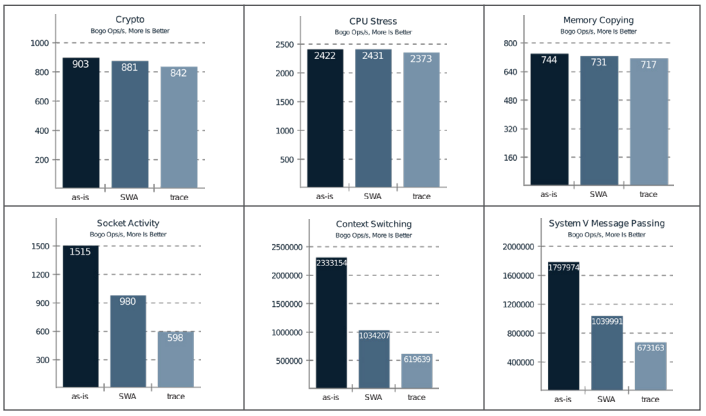


Figure 5. *Phoronix Stress-NG Benchmark Results: as-is is the system without tracing nor verification; SWA is the system while verifying Sleeping While in Atomic automata, and the trace is the system while tracing the same events used in the SWA verification.*

with automatic code generation and by leveraging the Linux tracing features to enable the kernel's online verification. The performance results confirmed that the automata's simple format translates into fast verification operations, allowing the verification of the system in production. Moreover, the ability to take action on the occurrence of unexpected events is a crucial technology to enable Linux's usage on safety-critical systems, increasing the need for such verification methods on Linux.

Additionally, we showed in the first article how to formalize the specifications of the real-time Linux kernel. Here, we have shown how to verify these specifications efficiently. With these results, we achieved the goal of ascertaining the logical behavior of the essential synchronization of the real-time kernel. However, the correctness of real-time systems does not depend only on the system's logical behavior but also on the timing behavior. The next article in this series will discuss how to leverage the model to extract a sound bound for the primary real-time Linux metric, the scheduling latency.

This article is a summary of the paper Daniel Bristot de Oliveira, Tommaso Cucinotta, and Rômulo Silva de Oliveira, "Efficient formal verification for the Linux kernel," International Conference on Software Engineering and Formal Methods (Cham: Springer, 2019).

Feature

Changing the world, one lesson at a time

Why teaching more teachers is essential to computer science education

by Matej Hrušovský

About the Author

Matej Hrušovský has been with Red Hat for more than 7 years, 5 of which have been spent managing the university program in EMEA. Aside from attracting new talent mainly from universities and schools, the core of Matej's job is to find and put the right people (from Red Hat and academia) in the same room together.

"Give a man a fish, and you feed him for a day. Teach a man to fish, and you feed him for a lifetime"—so the saying goes. There are two reasons you can tell this saying is very old. One, you've probably heard it a million times. Two, it optimistically assumes there's enough fish left to last a lifetime. For this cycle to continue on any scale, somehow you have to replenish the supply—the supply of fish, and the supply of available teachers.

Teaching is a critical activity for Red Hat Research, especially when it comes to mentoring student interns. As our efforts in this area continue to grow, we find we are victims of our own success: we are running out of available mentors. Especially with our resolve to retain a high quality of teaching, scalability was going to be an issue. The supply of Red Hatters can't last forever.

Here are stories of two people who are expanding Red Hat's teaching impact outside our walls.

MARTINA HAMANOVÁ—BRNO, CZECH REPUBLIC

Our story about Martina Hamanová begins with her choice to study mechanical engineering. At the time, the future of the industry didn't show a lot of promise, so she started from scratch and instead pursued a career in sales and accounting. Many years later, when her employer hit a crisis and she was laid off, a friend told her there was an opening at the school where she taught. Martina accepted the challenge and acquired the education necessary to allow her to teach, thus starting another new chapter in her life.

Martina Hamanová (left) has found that children are full of ideas, they love having fun, and they don't need to get it right immediately.

Her main responsibility at her new job was leading extracurricular activities after school for children ages six to nine. This kind of teaching differed from the focus on classroom topics and getting a good grade. Instead, Martina could do practical teaching and introduce children to the world around them. That could mean exploring the

four seasons outside, learning how to prevent injury, or keeping safe in traffic. Or it could mean building robots—and this is where her engineering background came in handy.

Children get very curious at this age, and they ask a lot of questions. Inevitably, having built robots and such, the subject of programming arose. Martina sought out PyLadies to learn the basics of programming so she could teach it to her curious students (see the article "PyLadies, Welcome to Open Source!" in this issue). Her son put her up to it, and despite her initial reluctance, she soon realized how much she enjoyed taking those classes. Whether or not she became a top-notch programmer, she could definitely learn enough to be able to introduce curious minds into the world of programming.

However, programming is one thing, and teaching it to children is another. This is what brought Martina to KiCo (Kids Coding Academy), organized by Red Hat. She first came with her class, and she was immediately shocked by the excitement of the children: they were so taken in that they didn't want to leave. Upon coming back from the workshop, it was clear to her that this was how she wanted to teach children. A few months later, she came to the same workshop, organized for teachers.

Martina has found that children are full of ideas, and they love having fun. They don't need to get it right immediately: learning is an iterative process that involves asking a lot of questions and making a lot of honest mistakes. Learning how a computer understands language then trying to use a precise set of commands to get their teacher to make them a real sandwich—and failing miserably—teaches them more about programming than memorizing scripting language ever could.

Teaching—like fishing or changing the world—is a long-term, iterative process.

Menachem Tsarfati—or Menny, as his colleagues call him—hasn't given up teaching despite his position as a Solution Architect Manager for Red Hat.

MENACHEM TSARFATI— TEL AVIV, ISRAEL

For our second story we will travel to the warmer climate of Israel, where the days are longer, particularly in Ra'anana, Tel Aviv. That's where Menachem Tsarfati—or Menny, as his colleagues call him—resides.

Menny is by no means a new face in academia. Even in his previous employment in a small integration company, part of his job agreement was to teach regularly. After accepting a traveling regional position, he still found the time to keep teaching. Now a Red Hatter, he works as a Solution Architect Manager—and still hasn't given up teaching.

His first long-term teaching commitment is with the Israel Defense Forces (IDF) Academy for Computer Science and Cyber Defense (also known by its Hebrew abbreviation, Basmach).

The IDF Academy provides training for soldiers to become professionals in computer science. Although the academy doesn't provide an official degree, that doesn't diminish its impact on the market. Many Basmach graduates end up leading large commercial companies in Israel.

Menny is involved in two major disciplines: microservices and big data and analytics. Microservices weren't very well known when the cooperation with IDF Academy started. They were the core of the first course, including development on top of OpenShift and its practical usage. Since then, more than thirty workshops have been held on this topic. Teaching students how to use Open Data Hub or to establish their first machine-learning-as-a-service is the focus of big data and analytics courses. These students could eventually end up being data scientists, data analysts, and big data developers. All of this ties back to OpenShift, which comes as no surprise given that IDF is one of Red Hat's biggest customers in the region.

Rather than wait until a scaling issue occurred, the academy began offering trainings for trainers to ensure an adequate number of instructors. These new trainers can now provide elementary lectures on microservices and OpenShift, while getting a technology update every quarter. This provides potentially unlimited scalability without relying solely on a small number of Red Hatters, making it possible to impact many more students.

Menny's second effort is based in the Academy College of Tel Aviv. Acting as a guest lecturer, Menny teaches microservices, architecture, and strategy once each semester.

Soon after he started, the academy's headmaster invited other lecturers to join the course, and many did. The partnership between industry and academia has a lot to offer both students and teachers. For example, students benefit from practical examples: taking a big monolith of a service and making it into small pieces, or implementing a database in a few minutes in front of their eyes. This tangible information helps them understand the implementation of the technology and see the big picture. Because the commercial market moves fast, in many cases industry is more up-to-date with technology. This is particularly true in applied practice, like implementing microservices. By contrast, academia is naturally strong in research. Both sides can bring these strengths to the table and benefit each other.

Teaching—like fishing or changing the world—is a long-term, iterative process. A very important factor for success is getting more people working towards the same goal. That way there's no need to lose momentum when internal resources start to run thin. Martina and Manny are just two of many examples of success stories demonstrating that training new mentors and teachers can spark a revolution in computer science education anywhere. RH RQ

```
const fetchingBlueprints = (state = false, action) => {
 switch (action.type) {
   case FETCHING_BLUEPRINTS:
    return true;
// We went from selling boxes of Linux®
   case FETCHING_BLUEPRINT_NAMES_SUCCEEDED:
    // if 1 or more blueprints, fetching is true because we're still waiting
on the contents)
    return action.payload.blueprints.length > 0;
   case FETCHING_BLUEPRINTS_SUCCEEDED:
 // to operating servers around the globe
 case BLUEPRINTS_FAILURE:
    return false;
  default:
    return state;
 }
};
const errorState = (state = null, action) => {
 switch (action.type) {
// to giving new life to virtual machines
   case FETCHING_BLUEPRINTS:
   case FETCHING_BLUEPRINTS_SUCCEEDED:
    return null;
   case BLUEPRINTS_FAILURE:
    return action.payload.error;
// to pioneering a new era in containers
   default:
    return state;
 }
};
// to powering over 90% of the Fortune 500*
const blueprintList = (state = [], action) => {
 switch (action.type) {
   case CREATING_BLUEPRINT_SUCCEEDED:
    return [
     ...state.filter(blueprint => blueprint.present.id !==
action.payload.blueprint.id),
     {
      past: [],
// to bridging every kind of cloud
      present: Object.assign({}, action.payload.blueprint, {
       localPendingChanges: [],
       workspacePendingChanges: []
      }),
      future: []
     }
    ];
// And we were able to do all this because
```

Our code is open_

Red Hat

redhat.com/ourcode

Red Hat

RESEARCH QUARTERLY

VOLUME 2:4

Feature

PyLadies, welcome to open source!

How did a group of three library students become part of an international force for promoting programming education? A Red Hatter who was there has the story.

by Petr Viktorin

PyLadies is an international mentorship group focused on bringing women into the Python open source community. In many places around the world, they organize meetups. I first made contact with them in 2012, when Lynn Root from PyLadies San Francisco visited Brno. Root wanted to start a PyLadies chapter here, but couldn't find women interested in discussing Python. She gave me a stack of stickers to give to PyLadies, should they organize. I gave the stickers to the first group of women I found that had something to do with Python in Brno: women who had organized themselves to learn programming.

While I did play a part in bootstrapping that group, and I still work with them, I'm not a member of the group. My role in the story comes from a different viewpoint: community programming courses. As with many open projects, the story involves many people who contributed and selflessly helped others. Here I'll only tell my part of the story. I will need to simplify and to leave many amazing contributors out, and I hope everyone involved will forgive the omissions.

HOW IT STARTED

PyLadies CZ are most known for organizing courses, mainly a beginners' course, which I've been teaching in Brno for about six years now. Let me explain how that got started.

I'd been co-organizing Python community meetups in Brno for around a year when a few students of library studies asked for help with their Python course. I took on the challenge and told them to consider bringing a few more classmates. All three were women, so I suggested they could call themselves PyLadies. My employer, Red Hat, kindly provided a space. Red Hat already had a process in place for organizing community events, and meeting rooms were available for evenings and weekends.

The students shared their homework and teaching materials with me, and I could immediately see why they needed help. Their course was designed to train computer scientists. Starting with the first lesson's motivational examples, the course assumed a deep love for mathematics, statistics, and puzzles. To future librarians, it didn't portray programming as something useful.[1]

1. Since then, the university has introduced a Python programming course for people not studying computer science, which, as I've heard, fixed the issues that got me started teaching. Thanks to Masaryk University in Brno, both for making that mistake and for fixing it!

Some of my first students saw tutoring as an easy way to get answers to homework questions. Those were disappointed—instead, I only gave hints and explanations of concepts. After about three such tutoring sessions (and the course's midterm exams), a lot of attendees left, but one came up with an idea to continue meeting regularly and go over the basics of Python and programming again, discussing them independently from their course. That sounded pretty fun to me, a software engineer.

The content was mostly up to them. I would provide a list of suggestions of what's possible, they would choose a goal, and I'd explain whatever they needed to reach it. Together we made a few web API clients, some games, and a plane ticket reservation simulator. Over time, people gradually left. At the end, one attendee remained: a teacher who convinced me to gather a new batch of women and start a PyLadies course, explaining the basics again.

THE SECOND RUN

One thing I realized as people were dropping out of the first study group is that explaining things to a smaller group of people is much easier than working with a bigger crowd. A small group is more focused, and if someone has an issue, it is often interesting to solve it with everyone involved. Larger groups don't have such luxury.

The new course, however, was not to be small. About twenty women signed up—enough to fill a conference room. Teaching so many people felt a bit scary. Then again, I told myself, the worst that could happen was that I'd waste a few hours of their time as they realize they're not satisfied. Perhaps the group would even shrink to a manageable size that way. I asked colleagues from Red Hat to help, and the idea to spend an evening a week with the PyLadies resonated with a few of them. Some had time to come to the lessons; one even set up an additional meeting for extra tutoring.

Since the most interesting projects from the previous study group were games, I decided to teach the attendees to write a game. I think that was—by luck—a good choice. Today my courses still feature games, despite criticisms about practicality. It's not easy to find a practical topic for general beginners: data analysis, system automation, website backends, algorithmic puzzles, or any other topic is interesting to some but boring or exotic to others. Games are impractical for everyone, but fun for most. And I'm careful to always mention that the skills learned on simple games are transferable to any other field.

Looking back, the course was quite imbalanced and chaotic, but I'm

I knew things like this happened in the open source world... I was blown away by the effect it had.

pretty confident that it was worth the money (0 Euros) for everyone, and worth the time for most. It was a reasonable first iteration.

WRITING DOWN AND OPENING UP

I needed to prepare what I was going to say, and as a relatively inexperienced speaker, the best way to do that was to write it down, word by word. While I would not follow the script exactly in front of the class, I'd have something to reference if I stuttered, and I could make sure the explanations made sense. I put my notes online so anyone who missed a lesson could read it. Since they were quite verbose, they turned out to be useful for self-study or for organizing a similar course. And since it makes no sense to limit the content of a free course to just enrolled attendees, I put them on GitHub under an open content licence. Anyone could read them, reuse them, or improve them if they wanted, in the typical open source way.

People from other cities took the materials and organized a course in Prague in 2015, then in Ostrava in 2016, and more cities afterwards. An attendee created a web page at pyladies.cz to list the different cities and courses. People started contributing to the published materials, first fixing typos, then reorganizing the course to suit different lecturers, and even writing entire new lessons.

I knew things like this happened in the open source world, but this was the first (and only, so far) wildly successful project I started. I was blown away by the effect it had.

READING UP

The next episode in the story came after I saw a recording of Greg Wilson's talk at PyCon 2014 in Montréal, "Software Carpentry: lessons learned." Software Carpentry is an organization that teaches programming to scientists. They do a very good job at it, incorporating pedagogy research and openly sharing all their materials and know-how.

I read the literature recommended in the talk and incorporated many ideas Greg mentioned, like giving students red and green sticky notes with which they can quietly indicate if they are stuck or, respectively, finished with a task. Since then, Greg wrote *Teaching Tech Together,* an openly available book for anyone who wants to run community courses. I heartily recommend it.

DELEGATION BY RESIGNATION

Teaching and writing content is quite time consuming, but I enjoy it. There are other aspects of organizing a course that I did not enjoy as much: selecting attendees, recording feedback, or reserving the meeting room. Around 2016, I announced that I would not organize the next run. If there was to be another course, PyLadies Brno would have to organize it themselves. I could still teach, and be available to help, but only if asked.

Thankfully, organizers did step up. Some PyLadies liked the community and wanted to continue meeting like-

minded people. Some felt that it would be fair to "pay" for the course with their time. Some wanted experience organizing events. Some wanted an excuse to be at the next course, so they could go over everything once again.

With courses no longer restrained by a single tired organizer, they flourished. The new organizers started going for dinner and drinks at a restaurant after each lesson, leading to more new friendships—and a lowered drop-out rate. They came up with a better process for selecting attendees or for giving feedback. I did not agree with all the changes, but since I was no longer an organizer, I gave suggestions and then kept silent.

Having delegated the organizing to others, I'm essentially doing only the role I like: preparing lessons and teaching. The nice part is that nearly everyone involved also does only the roles they like. And when someone stops liking a role, there's usually someone else to fill the empty space. From this point, I started seeing PyLadies Brno not as the courses' attendees, but the organizers. The attendees often include a guy or two. I, with my beard, am just an external lecturer. It's the PyLadies who make it happen.

PROGRAMMING VS. COMPUTER SCIENCE

I teach programming, rather than computer science. I skip algorithm design and all kinds of theory I learned (and enjoyed!) at the university. Sometimes, that leaves people worried. A century ago, each car driver also needed to be a mechanic. A few decades ago, each programmer needed to be a mathematician. Neither is the case today. Just like you don't need to know how to replace a brake pad to drive, you also don't need to implement Quicksort to automate downloading spreadsheets. Of course, just like we need car mechanics, computer science is still relevant. We just don't need everyone to be a computer scientist. And universities already do a good job teaching the interested people.

A century ago, each car driver also needed to be a mechanic. A few decades ago, each programmer needed to be a mathematician. Neither is the case today.

I would actually be OK if my students forgot all programming skills and were only left with an idea of what programs can do. That way, next time they find a very repetitive task, they know it's possible to automate it. Even if they forgot how, they will know that a programmer can automate the job for them, just like they know to look for a lawyer when they need to sign an important contract.

GOING ONLINE

During the years I spent teaching, I always resisted putting recordings of the lessons online. It's not the right format: the explanations and pauses are tailored to a specific audience. Also, if the participants know they're being recorded, they are much less likely to ask questions.

In 2020, we can no longer meet physically, so everything is now recorded. And since questions are mostly asked in text and there's no need to make the videos private, I began streaming the lessons on YouTube. There's still interaction with the audience, so I don't think they're good video lessons on their own, but they do get more views than participants. My goal is to improve the learning materials, which everyone else can then use to learn or teach from. The live class is the better way to test the current iteration on live people and gather feedback.

I think such testing could apply to any set of instructions. Next time you write a tutorial for a piece of technology, consider finding people from your audience and doing a community workshop or course to test it out. Chances are, you'll learn a lot about how people understand whatever you're trying to explain.

Or you might find a great way to help a community grow.

About the Author

Petr Viktorin leads the Python Maintenance team at Red Hat, where he contributes to Python and its integration into Fedora.

Column

Shared knowledge or private IP? That is the question

RHRQ interviewed Idan Levi, the Research Interest Group leader in Israel, to get his take on how university research intersects with the open source approach, from datasets and collaboration to security and data privacy.

RHRQ: You lead Red Hat Research in Israel. Based on your experience, what are some of the benefits of sharing software code, environments, and data that produce research results?

Idan Levi: First and foremost, the academic world started as open and shared their research with external partners. This is the pillar of human knowledge. So in order to promote research, we must be open. We must be open so that others can collaborate and add their insights. We must be open to criticism so that we understand when we are wrong. And it really connects to the core values of Red Hat: working openly with the diversity of the community.

We must be open so that others can collaborate and add their insights. We must be open to criticism so that we understand when we are wrong.

As you rightly mentioned, it's not just about the code. It's also about the data—the data that we produce, the data that we use. In order for us to do research, especially around AI and machine learning, we need tremendous amounts of data. And getting quality data that reflects event time series and different parameters can sometimes be difficult. Without access to this amount of data, it is difficult to meaningfully advance research. Researchers working in open source are able to access data that they otherwise would not have access to. Our work, for example, within OpenInfra labs helps to deliver open source tools to run cloud, container, AI, machine learning, and edge workloads efficiently, repeatedly, and predictably.

RHRQ: In recent years, a lot of companies with huge investments in software development have gotten involved with open source research. How does this affect the development of open source software?

Idan Levi: I think this speaks to a trend that we have seen over the years. Take a look at the field of big data and data processing. These fields are heavily dependent on open source research and contributions. Hadoop and Spark are good examples of open source utilities that come to mind to enable all of that.

We are also seeing a lot of partnerships that are built around these projects. By moving their research from being proprietary and opening up their use cases, these companies are advancing innovation in a scalable way. Through collaboration with upstream communities, they can get feedback and better understand what challenges they need to solve. And I've witnessed this. We recently

had a great accomplishment where one of our research partners approached us and said, "Hey, we have a great technology to enable data skipping for Spark. And we even incorporated it to our cloud offering. But nobody outside really uses it and understands the benefits. If we are to collaborate, we'll open source it."

This was a very interesting conversation. The way it works is that you first open source the technology, get some feedback, change it a bit, and only then do we use it. The researchers and the engineers understood how to do it. It took a little longer for the managers. But once they got to meet the community, it became apparent to them that it's not about just fixing a bug. We also needed to consider how plugable the technology is as well as how best to maintain it. This kind of thinking certainly makes the solution much better.

With open source research, innovation happens in the upstream. This process helps to address customer problems in a more efficient way. It's not just about a solution that looks for a problem anymore. It's about the synergy of problem solving and evolving, together.

RHRQ: How would you say intellectual property (IP) is handled in the open source research realm?

Idan Levi: That's a challenging question. I think with the rapid pace of change today you cannot really hold on to a particular piece of IP for too long because someone will try to go past it. Someone will either recruit better people, wait for the patent to end, or try to work it in another way. In my opinion, holding onto a single IP does not really serve as a unique advantage over time, especially with the history of long patent wars over the years. It actually slows down innovation. The real value comes from collaborations, partnerships, and exchanging ideas.

Idan Levi leads Red Hat Research in Israel under the Office of the CTO organization. He has over fifteen years of experience in the industry, including the management of R&D teams with extensive knowledge of data systems, design, and architecture.

Bear in mind that customers want to work with companies toward a certain vision they share of the future, not just because they are locked in to an architecture, solution, or platform. Vendor lock-in makes it difficult for customers to integrate with some of the other systems they own from a different provider.

I think the world of IP that locked you in is changing. In this new ecosystem where collaboration drives innovation, we find that more partners and customers are viewing IP as a hindrance.

RHRQ: One big concern expressed by partners or customers who are just getting started with open source is data privacy and security. How can they still get involved with open source research?

Idan Levi: There's both philosophical opinions as well as proven studies that show open source is generally more secure. I can also understand and relate to the concern about open source and data privacy rather than open source and security. With regards to open and secure, you have many more people looking at the code base and therefore it is easier to test and fix issues. It is much easier to file a pull request rather than contact a vendor through their support channels to report a bug. And in terms of data privacy, there are also systems that allow you to publicly share information about a dataset while withholding information about individuals in the dataset.

"...the world of IP that locked you in is changing."

Research project updates

Faculty, PhD students, and Red Hat associates in the Czech Republic are collaborating actively on the following research projects. This quarter we once again highlight collaborative projects at Masaryk University (Brno), Brno University of Technology, and Czech Technical University (Prague). We will highlight research collaborations from other parts of the world in future editions of the Red Hat Research Quarterly. Contact academic@redhat.com for more information on any project.

PROJECT: AUFOVER (Automated Formal Verification)

ACADEMIC INVESTIGATORS: Honeywell: Mgr. Tomáš Kratochvíla; Paradise: Prof. RNDr. Jiří Barnat, PhD; Formela: Doc. RNDr. Jan Strejček, PhD; VeriFIT: Prof. Ing. Tomáš Vojnar, PhD, and Ing. Aleš Smrčka, PhD

RED HAT INVESTIGATORS:
Kamil Dudka and Ondřej Vašík

In 2019, we offered Symbiotic and Divine as RPM packages and developed utilities to convert their output into a unified, machine-readable format. We then started to see unexpected changes in the output of formal verification tools as the tools are updated. This observation drove us to create our own test suite, named aufover-benchmark, where multiple tools are exercised by the same set of tests. This in turn helped us discover previously unknown bugs in the formal verification tools. These bugs were reported to developers of the tools, and many of them have been fixed.

Now we are moving the automation to the next level. Our ultimate goal is to run formal verification tools on unmodified source RPM packages. Unlike static analyzers, where we instrument only compilation of the source code, we now need to instrument the execution of binaries produced by the build in addition.

At the same time, we want to interfere with the testing frameworks used by RPM packages as little as possible. To tackle this problem, we developed csexec—a dynamic linker wrapper. The wrapper can be used as an ELF interpreter while linking binaries during the build of RPM packages. Then we can transparently choose which formal verification tool should be used to instrument execution of the binaries while running test suites embedded in source RPM packages. Although the original motivation for developing csexec was to run formal verification tools, the wrapper can easily be used to run dynamic analyzers (such as valgrind or strace) on unmodified RPM packages, as shown in the following quick demo: https://github.com/kdudka/cswrap/wiki/csexec

PROJECT: PatrIOT—IoT Testing Framework

ACADEMIC INVESTIGATORS:
Miroslav Bureš

RED HAT INVESTIGATORS:
Štefan Bunčiak and Miroslav Jaroš

PatrIOT has resulted in three submitted US patents by Red Hat and three submitted CZ patents by FEE CTU (Faculty of Electrical Engineering, Czech Technical University). This project is pushing state-of-the-art borders in several R&D areas. As the IoT industry is growing, the project has hit perfect timing to fill the market gap in the field of IoT testing. The testing framework was very well received on the market by important enterprises—namely, Skoda Auto, Rockwell, Siemens, and Electrolux—who provided positive feedback. The grant for this project from TACR (Technology Agency of the Czech Republic) ended in 2020, but the project now continues as self-funded until the next call for grant applications.

PROJECT: Innovation Scorecard

ACADEMIC INVESTIGATORS:
doc. Ing. Ondřej Žižlavský, PhD, Eddie Fisher, and Tetyana Shpilka

RED HAT INVESTIGATORS:
Marcel Gazdík, Vojtěch Sokol, and Tomáš Meszároš

Innovation Scorecard, originating at the Faculty of Business, was a rather unconventional project for a Red Hat partnership. Prior to this project, Red Hat Czech was mostly concentrating on cooperation with faculties focused on computer science. This project provides a methodology to drive innovation in project management. In 2018 it was funded by TACR for three years. After two years of working with multiple teams in Brno, a lot of case data has accumulated, resulting in publications and mentions by the Project Management Institute (PMI). Innovation Scorecard is now heading into its third year with Red Hat, and its creators are planning to publish an ebook with the certified methodology.

PROJECT: Vega Project

ACADEMIC INVESTIGATORS:
Gabriel Szász

RED HAT INVESTIGATORS: Nikolaos Moraitis, Zdeněk Švécar, and Filip Hubík

The project is using OpenShift as a platform for execution and monitoring of scientific computations. An OpenShift-based framework supports scalable parallel open hybrid computing and enables easier distribution, scalability, and flexibility of already-written code that cannot be compiled on a supercomputer or parallelized otherwise. In addition, the decomposition of a complex pipeline into simpler subunits provides better maintainability and simplifies debugging. The project is currently challenged by lack of resources. Options for providing more hardware (and thus computing power) are being explored to allow the project to move forward.