I/O VIVAT

YEAR 38
NUMBER 1

Numerus fixus
A new age of Technical Computer Science

The History of Pandora
From Assassination Game to Extraterrastial Encounter

ENIAC Thesis Awards
Matthijs Souilljee, Rick de Vries, Chakshu Gupta & Remco Abraham

Facebook Crypto
The story of 'Libra'

AI Regulation
Regulate, before it's too late

And more...
Columns & interviews
Company visits
Guest writers
Prodrive Puzzle

Inter-Actief

//Colofon

Volume 38, number 1,
February 2023,
ISSN: 1389-0468

I/O Vivat is the scientific magazine of I.C.T.S.V. Inter-*Actief*, the study association for Technical Computer Science, Business Information Technology and the corresponding Master's programmes at the University of Twente. I/O Vivat is published two times a year with 1900 copies.

// Chief Editor
Emma Sloot

// Executive Editor
Niels de Groot

// Editors
Xanti Lizanzu, Niels de Groot, Yoeri Otten, Ruben Groot Roessink, Sven Mol, Emma Sloot

// Guest writers
Rick van Galen, Maxim de Leeuw, Oliver Davies, Kimberly Hengst, Matthijs Souilljee, Nick van Apeldoorn, Vadim Zaytsev, Rick de Vries, Chakshu Gupta

// Special thanks
Leo de Penning, Ramon Ankersmit, Roeland Krak, Pieter Staal, Gerrit-Willem Smit, Wouter Suidgeest, Wallace Ugulino, Patrick Broeckhuijsen, Maria Khovanskaya

For questions, comments or suggestions I/O Vivat can be reached via e-mail at iovivat@inter-actief.net, by phone at 053-489 3756 or by mail: Study association Inter-*Actief*, PO Box 217, 7500AE Enschede

// Printing
Drukkerij van den Bosch & Fikkert

I/O VIVAT

//Editorial

Dear reader,

In front of you is the new I/O Vivat, the association magazine of I.C.T.S.V. Inter-*Actief*, and the magazine that we gladly share with our alumni through alumni association ENIAC. We can only wonder what it feels like to receive a surprise I/O Vivat on your doormat, as we are the ones that send it, kind of ruining the surprise for us, but we hope it has been a pleasant one after all this time. As you may know, the past few years have been… weird. However, it is not just that. Creating a magazine is harder than you think.

It starts with finding the right content: who will write something and what will they write? An idea is born, but an idea is merely that: an idea, in your head. For an article, it seems convenient to have something on paper, so you start writing. You finish writing. A review. A second review. Fitting it all together, waiting for a late submission, contacting partners, etcetera. The timing is especially crucial because if something is not ready it will hold up all the other items, if you decide to publish you will be short on pages, and if you postpone some items may become irrelevant. Creating a magazine is like cooking: if you do not make sure the ingredients are ready at the same time, there is no meal.

As we write this, we have both been chefs in this kitchen for longer than we had ever expected. Niels even already started his first activism at the association at the I/O Vivat in 2017. The thing is that after a while in the kitchen, cooking the same food as always is no longer as fun. You get stuck in a pattern. You can be demotivated by those hungry for more, while your appetite is fulfilled. The best option at that point is to take off your apron and hand it to someone else, which is exactly what both of us will be doing after this issue of the I/O Vivat. Fortunately, the apron will be worn by two new chefs who will be cooking up something rather nice rather soon. We wish them all the best in their efforts!

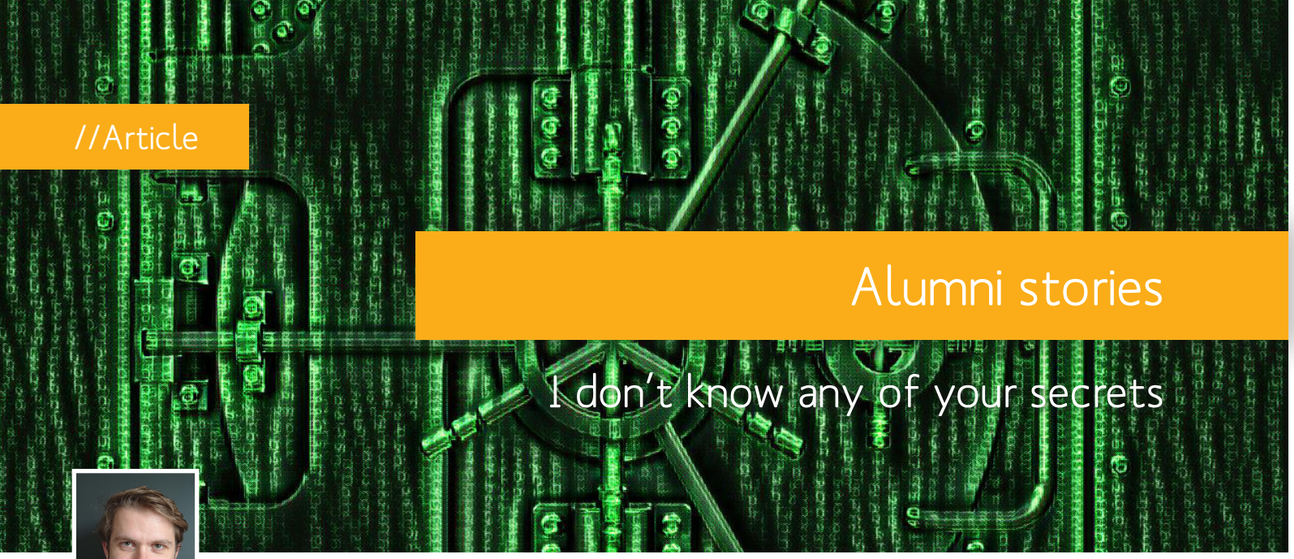
It has been an honor to shape this magazine, in one way or another, over the past few years. We hope it has provided you with ample knowledge and entertainment, and we genuinely hope that the work we put into this magazine has been worth your reading every time again.

Thank you, and until we meet again,

Emma Sloot
Chief Editor I/O Vivat

Niels de Groot
Executive Editor I/O Vivat

//Contents 38.1



//Article

Alumni stories

I don't know any of your secrets

by: Rick van Galen
Guest editor

What does "security engineer" really mean? Being a security engineer really varies depending on what a company values and what role you're given.

In most organizations, security engineers have a strong infrastructure focus. After all - it's your IT infrastructure that keeps you going, and that's where your attackers get in. If your organization focuses on delivering a reliable service with a complicated infrastructure with lots of moving parts, then there is where the complexity lies.

That focus is slightly different in a pure product company. Any password manager for example is "just a glorified key value store". While of course the reality is **a little** more complicated, a significant amount of complexity lies in the workings of the product. You can keep your infrastructure conceptually simple, and work on the complexity of the product. Therefore, my focus as security engineer skews strongly towards product questions. How do I reduce the information I know about customers? How do I keep bad actors out of our dependencies? How do I make our product more user friendly within the existing security principles?

The thing that unites good security engineers in my opinion are that they are the **department of YES** instead of the department of NO that security folks usually end up being. Security engineers develop technical solutions - and yes, also **process** solutions - so that all engineers can keep building a great product or service.

Why did you move from attack to defense?

I enjoyed being an ethical hacker. There's something incredibly satisfying about being able to solve puzzles that others didn't want you to get solved. Chaining weaknesses together - technical weaknesses, process weakness, or weakness in human psychology - makes for a very diverse job. I can recommend it to anyone.

The thing that got in my way is that I'm too much of a big picture guy. Things that motivate me is making some difference in the lives of everyday people, and working for companies that I believe in. If you're a hacker-for-hire, you sometimes do that, and that's awesome. But realistically speaking that is not where the bulk in demand is. After I solved a bunch of puzzles and showed a customer the flaws in their security, I never stuck around to see the difference that made. And moreover, I had too little choice in picking what kind of puzzles I could be solving, or the amount of time I could spend solving them.

The defense side of things is less romantic but much more charming. You get to sit down and focus on bigger stories - on building better resilience, faster processes, and product that is easy and safe. It gives me more satisfaction being key in protecting your Coolblue logins (at my previous job) or in protecting the secrets of 1Password users everywhere, including friends and family.

Why did you move from Utrecht to Toronto... in 2020?

It's not as bad a timing as it sounds - truly! While southern Ontario is not in any better state than the Netherlands, maintaining social life everywhere sucks in a pandemic. So we're not losing any time from that perspective. Outside of social life moving in a pandemic is not as hard as you might think, if you're one of the lucky ones that has an easy time finding employment. Any socially distanced time can easily be used to get settled in a country from a practical perspective - doing your groceries, paying your taxes, understanding local government, and understanding differences in work culture. Not glamorous by any means, but these are not glamorous times irrespective of where you are.

What certainly helps is working for a company that has a very strong remote working culture. When Covid hit, very little changed for 1Password which has been a remote company since their founding. Having all your internal communication already organized around remote work assumptions makes a huge difference in way you get into your work. I'm personally amazed at how truly effective you can be in your work, have a lot of flexibility in work-life balance, and how you can build great stuff, while your manager is thousands of kilometers away.

What makes protecting secrets and passwords different from "regular data"?

The threat model around secrets and passwords is so different from "regular data", that it shifts the focus of where you focus your attention a lot. Any password manager worth their salt can choose to know **as little as possible** about their customers. It's the same in my situation - I absolutely don't know anything about the contents of your vaults, and the only things I can learn (with a bunch of caveats) are things I **need** to know - how you pay for your account, and how many vaults and users you have. There is never any reason for the company to need access to your vaults whatsoever, and therefore we chose to make ourselves not be able to.

At the same time, the secrets a password managers stores are the most personally sensitive information there is. That makes me extremely nervous about protecting *both* your vaults when they reside encrypted on our servers but also all your interactions when you decrypt your own passwords client side. Our security has to be truly end-to-end.

You would not leave your keys dangling around, so why would you leave your passwords vulnerable?

> "Any pasword manager worth their salt can choose to know as little as possible"

What are some of the bigger problems you're solving?

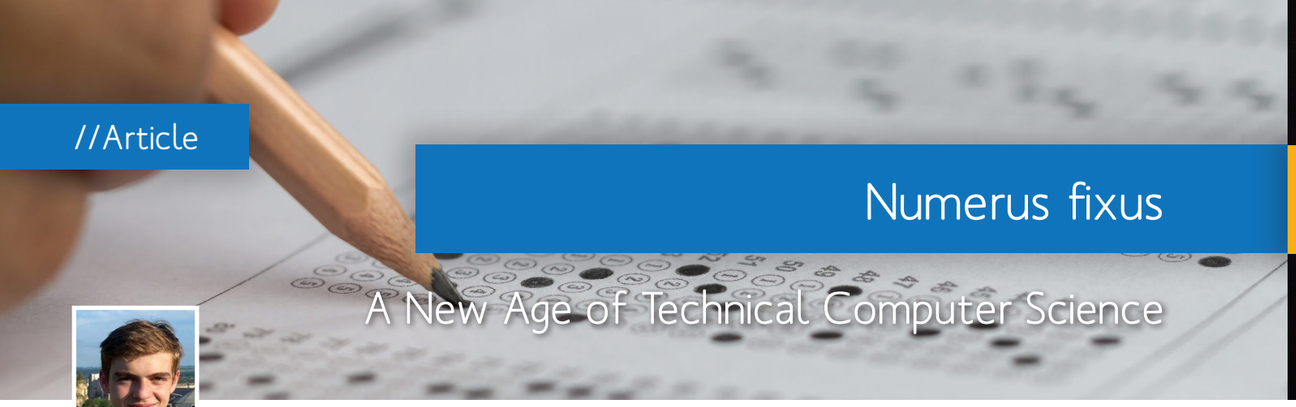
A major change the company is undergoing now is that we're transforming from being a consumer only password manager to being *both* a user friendly consumer product, as well as a user friendly secrets manager for IT infrastructure. This doesn't change much about the equation of the secrecy of your data that needs protection, but it does come with new challenges in expectations about access control, infrastructure integrations and event data that brings. Since we're the first password manager to make a big move in this area - acquiring a Delft based startup in the process - we still have all the problems to solve there.

One unique difficulty a password manager has is keeping up an effective bug bounty program. You might expect that password managers are a big target for security researchers, but that's only half correct. When you build a product that has so many layers of defense that you'd need custom tools to pry open, that becomes a significant barrier of entry. Some people cross this barrier of entry, but security researchers like most people are more inclined towards targets. So I find myself in the position that I'm writing and publishing security research tools for our own products - just so I can have as many eyes as possible on our stuff.

An intellectually very exciting problem I'm solving has to do with **cryptographic agility**. The cryptographic protections we have on passwords are great, but we can't expect them to be great forever. So we need to make sure our protections are **agile** - so that we can use the latest state of the art to protect your data at all times. But how do you do that when you share vaults with, say, the marketing team or your parents? And how do you do that when you can't be sure everyone else is running the latest app version? How are you eventually getting quantum computer resistant cryptography into all **that**? We think it's possible, but it's still a work in progress!

About Rick

Rick studied Computer Science at the University of Twente and Information Security at the former Kerckhoffs Institute. He was on the 31st Inter-*Actief* board in 2009-2010, was editor-in-chief for the I/O Vivat for a while, and graduated in 2014. After graduating, he joined KPMG where he was an ethical hacker, before joining Onegini where he lead product security. In 2020 he moved to Toronto, Canada where he joined 1Password as a security engineer. After his adventure overseas, he is currently back in The Netherlands.



//Article

Numerus fixus

A New Age of Technical Computer Science

by: Xanti Lizanzu
Editor I/O Vivat

Do you remember applying to Technical Computer Science? If you do not, it is probably because there is not much to remember. Some mouse clicks and on the first of September, you were officially a computer science student. Last year, students had to put in some more effort. Why? Because Numerus Fixus has (after much effort of the faculty) been introduced to the bachelor in Twente.

The change has caused a slight disruption in the balance of the settled world of Technical Computer Science. While Technical Computer Science in Eindhoven and Delft, and Computing Science in Groningen already had a selection, Twente was still an open program. Students who enter those selection processes used to use Twente as their backup, but this is not the case anymore. Before the introduction of Numerus Fixus, I discussed with program director Vadim Zaytsev what the change would look like. This article is a retrospective analysis of that discussion.

The idea of introducing Numerus Fixus was not revolutionary. Computer science is a notoriously growing field and the number of applications reflects this growth. In 2021–2022, the last year before the introduction of Numerus Fixus, 418 students started on the first of September. Much was done to support the growth, "but there is still a limit on how much you can grow without sacrificing quality and without sacrificing the personal touch." according to Vadim. This was the main reason to introduce a limit on the number of new students, together with the fact that it is preferred that Twente is somewhat in sync with Eindhoven and Delft. In retrospect, only 380 out of the 485 at the entrance examination accepted their invitation to the program.

To understand the Numerus Fixus, it is important to know what it actually consists of. A motivation questionnaire meant to filter out the unmotivated students is the first part of the process for new students. The next step for them to take is the actual test, consisting of three components: logical thinking, mathematical thinking, and algorithmic thinking. For the last one, it is important to note that it will not contain any actual programming questions. The university is interested in the students with the highest potential, not with the most experience. The students with the - in the eyes of the university - highest potential can then claim their place.

To construct the test, Twente peeked over at Eindhoven and Delft. In general, a lot of decisions were made by consulting other universities, also including Groningen. Still, much thought was put into how new students actually should be ranked. Maybe you noticed that school grades are not included in the ranking. No evidence supports the fact that high grades make good students, Vadim explains. Moreover, grades are differently valued in different countries. It was thus decided to rank only on test results.

The results of the first implementation of the Numerus Fixus were satisfactory, according to a member of the TCS Programming Committee. A complete website, transparency about the process, and a shared email inbox used by the candidates were the aspects that went well in the communication. For examination, proctoring was not able to be implemented, but the questions were unsolvable by using search or answer engines. The split-up between an on-campus and online examination were also deemed okay. All in all, the next Numerus Fixus will not undergo any major changes next year.

So Numerus Fixus is here and we have witnessed the introduction of a new age of computer science in Twente. The process will most likely be dynamic for the next few years. "Maybe we will grow to 500, 600, or even 1000 students," according to Vadim. For upcoming students, Vadim can assure you of one thing though: do not worry. An entry exam may seem like a big deal, but he ensures that it is not. The selection process is not there to make life difficult but to keep that beloved personal feel. Ironically, to conserve the beliefs of Technical Computer Science, its very core needs to be changed. And beware, this change will probably not be the last.

Starting September 2022, the study Technical Computer Science at the University of Twente has a numerus fixus. To find out more our new editor Xanti Lizanzu researched the implementation of the new admission requirement and talked to the programme director to hear his views on this change for the programme.

//Column

Multidisciplinary

Creative Technology / Electrical Engineering

by: Maxim de Leeuw
Master student Electrical Engineering

Hello everyone! My name is Maxim de Leeuw, and I am 25 years old. I have been asked to write for the multidisciplinary column of this magazine and talk about what I have done so far in my time at the university. I started studying at the UT in September of 2016 to study Creative Technology (from now on abbreviated to CreaTe), as the broadness of the CreaTe allowed me to explore my interest whilst the projects also gave hands-on experience. In the first two years of CreaTe, I learned a lot of new skills, especially on the disciplines of programming and design. The study also had more branches of disciplines, being electrical engineering, applied mathematics and business.

The second year was already focused on specialization and every student had to choose whether they wanted module five to be focused on the hard sciences (Smart Technology) or more on game design (now called Interactive Media), and I chose for Smart Technology as the hard sciences were way more interesting for me. The business module (M7) was also one I enjoyed a lot and I have been a TA for that module from the moment that I could.

In the last module of the second year, the choice made in module five came back around and decided which was the "elective" course in that module. As I chose Smart Technology, I could follow a biosignals and engineering course. I also thought this was a very nice course, and it awakened a new interest. This practically helped me to realise what I wanted to do for my Master study, but beforehand, I wanted to do a board year at my study association S.A. Proto.

My interest in money and business, which was already outed in the business module, again showed its head and I was able to spend a year as the treasurer of the association. After a wonderful year, I continued studying and chose to do the premaster Electrical Engineering in my minor space. It was a lot of work, as although I had an interest in mathematics, I still lacked a lot of knowledge. It took me a bit longer than I was supposed to, but in the end, I was able to finish the premaster and graduate from CreaTe. I feel like I became a real Creative Technologist, with having skills in all disciplines, but not being the top of my class in any discipline.

In September 2020, I started my master's in electrical engineering and chose for the specialisation of Neurotechnology and Biomechatronics. The courses that I am now following are mostly shared with students of Biomedical Engineering, and are focused on the discipline of signal processing, biological system analysis and neuroscience. Next to that, there are some mathematical courses, in which MATLAB is my biggest "frenemy". Yes, as someone who now has some experience with programming but who still is not very fluent in its languages, it is sometimes hard to use it, but it is also a programme with a lot of possibilities and functionalities to help visualise the biological signals.

I hope to finish my courses this year, so that I can start an internship to put the learned skills in to use. I wish to intern at the R&D department of Roessingh rehabilitation centre, as I have a personal connection to it and envision myself working in such a place at a later stage of my life as well. I hope I can put my creativity and broad knowledge into use later, and become an engineer that can help people all life long!

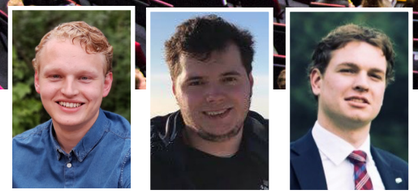
About Maxim

Maxim de Leeuw, originally from the nearby town of Borne, started his studies at the University of Twente in 2016. During his studies he has been active at study association Proto including his board year as treasurer (2018-2019). He has also been a teaching assistant and is currently one of the friendly faces you may encounter at the Servicedesk in the Hal B building.

//Article

Pandora

From Assassination Game to Extraterrastial Encounter



by: Niels de Groot, Yoeri Otten & Ruben Groot Roessink
Editors I/O Vivat

Recently, the 23rd edition of the weeklong puzzle/assasination game Pandora took place, renewing our interest in the humble beginnings of Pandora: where did this amazing event come from? We joined up with several old (and new) committee members to share their stories about the game we all love. In this article, we shine a light on Pandora's beginnings, its (r)evolution over time and its current set-up. For those who have never heard of Pandora or need a refresher, please take a moment to read the sidebar 'What is Pandora?'.

In the beginnings (Leo and Ramon)

Pandora first saw the light of day in 1998. Four committee members decided to organize the first Pandora under the title Pandora Assassination Game and with a James Bond theme. Leo and his fellow committee member were inspired by the Taste Assassination Game ('TAG') as organized by student assocation Taste. The concept was easy, a weeklong of shooting each other in combination with some (very difficult) puzzles. The vibe of the movie Gotcha!, about a campus-wide paintball competition at the University of California Los Angeles, best described the experiences of the first participants with Pandora. Although, it is not really known whether Taste was inspired by this movie when creating the TAG event.

The first Pandora was mainly characterized by the sheer ignorance of participants to this new event. The first six teams gathered after receiving a secretive message to meet at an abandoned building site. After some time a large black Ford would drive towards them. Then, the four committee members stepped out fully suited up and with black sunglasses, as if in a James Bond or Mission Impossible movie. The committees during this time consisted of several over-eager members using the (now infamous) Member's Initiative to organize the first editions of Pandora. After some editions the organization of this event transferred to the aXi (general activities committee).

As the first Pandora was not discussed with either campus security or with the faculty/university, and as participants were still trying the limits of this new event, several attacks during lectures resulted in a ban for all Pandora participants from lecture halls within the first 24 hours. Screaming 'Daar zit ie' ('There he is') during a lecture certainly did not help in preventing this ban. The committee members then also actively participated in Pandora, at least the fighting part. Before Pandora, the gameplay regarding the puzzles was already thought-out and during the week only the script had to be followed. Leo recalls that he was locked in his room for half a day, as two participants were camping around his front-door. He only managed to escape by closing all the blinds and crawling out the window of his toilet.

Figure 1: Puzzles are located at spots on campus nowadays, but in the beginnings, more of Enschede was also used.

The first edition(s) of Pandora were Enschede-broad events, meaning that puzzles were located all around the city and the campus. Puzzle locations included the Van Heekpark and Station Drienerlo (Station Enschede Kennispark, ed.). Each day was correlated with a single puzzle consisting of multiple steps, meaning that you had to physically move throughout the city to solve the next part of the puzzle. These puzzles often took the entire day to solve, also not helped by the fact that the internet was not a big thing then, although you did not puzzle for the entire night back then. The committee had to actively give hints to participants to solve the puzzles to get teams to 'accidentally' meet at a puzzle location. According to the participants and committee members from that period, the puzzles had a rather 'simple' set up, meaning that the same kind of tricks were repeated.

> "Several attacks during lectures resulted in a ban for all Pandora participants from lecture halls within the first 24 hours"

The fighting during Pandora during those first editions was also relatively simple. Hitting one another resulted in a single point being awarded and the team members had to actively report these results to the committee. The committee then manually counted all the points in order to determine the winner. The participants received weapons that are similar to today's Nerf guns, albeit with a range of about half a meter, so battles were mainly concluded close-range.

Digitalization (Roeland)

During Roeland's tenure at the University of Twente Pandora had already been organized for 10-15 years and had changed to a campus-only event. When asked what he saw as the main difference between his first time organizing the event and his third time organizing the event, he mentioned digitalization. During his first Pandora, kill codes had to be physically turned over to the committee at the start of each next day. During his last time organizing, a PHP-based website had been erected for entering kill codes, and thus keeping score, real-time.

In 2010, organizing Pandora and concurrently competing in the fighting part was already out of the question. This is also the time period where Pandora became its own committee, as opposed to being part of the aXi. Organizing Pandora was a six month ordeal, where the committee, while remaining secret, had to create more than 32 puzzles. During the Pandora week, the committee got a location at the university for an entire week and a committee member was always (24x7) on call for either requesting hints, or settling disputes between participants about who 'killed' who first. Although Roeland's effective method was to usually conclude that, since he was not there, the participants killed each other and should give the kill codes to each other.

'Days' were comprised of about 8 puzzles each day, for 4 days straight, excluding some bonus puzzles. Teams were awarded a time bonus for solving all 8 puzzles as soon as possible. Teams thus often puzzled till deep into the night. The committees experimented with new puzzle ideas with the technologies available at that

Figure 2: The pool noodle has been a trustworthy Pandora item for some time already.

What is Pandora?

Pandora is a yearly weeklong puzzle & assassination game organized by a secretive committee. It has continuously been organized since 1998/1999 by members of Inter-*Actief*, with only one year where it unfortunately could not take place due to the Covid pandemic. The goal of the game is to earn as many points as possible with your team, consisting of at most six members. Points can be obtained by either solving puzzles or by 'killing' members of the other teams. Several weapons enabled the fighting part over time, including blow darts, Nerf guns, pool noodles and starfish-shaped stuffed animals used as throwing stars. Every year, the committee develops a story line based on a general theme. If you want to find out more (including a huge set of old puzzles), take a look at: iapandora.nl

time, including puzzles that used VR/AR apps. Having this many puzzles also meant that the puzzles had to be tested thoroughly before Pandora. The committee set up a testing system, where testers could try to solve a puzzle and then click the location of the puzzle on the campus map to determine the complexity of the puzzle. Before this, testers often asked for a hint and then, after hearing the hint, concluded that the puzzle was easy enough. Making the puzzles harder to spot, by using black tape and a black background is one of the improvements from during this time as well.

Weapons included the Nerf guns (short to medium-range), Pool noodles (very short range) and Blow darts (long range). One specific remark is regarding the blow darts. These consisted of PVC pipes with paper darts, the latter becoming a true craftsmanship for some of the teams. These darts were created from (news) papers and sometimes the tip was hardened, using adhesive tape, for an even better range (sometimes distances of 30 meters were achieved). Several incidents of people having one of these darts stuck in their leg or stuck between their eye and eye socket have been recorded. Luckily, these hurtful and unsafe weapons have since been upgraded.

Now (Wouter, Gerrit-Willem, Pieter)

The current set up of Pandora has stayed the same in the last couple of years. The event still consists of 4 days (starting Monday evening and concluding

Figure 3: Blow darts have been known for their casualties. Luckily, the sharp arrows have by now been replce with nerf darts.

Friday evening) and is still campus-only. Several safe zones exist in which Pandora-related activities are not allowed to keep the disruption to the lectures to a minimum as well as ensure no suspicious behavior occurs surrounding some of the buildings on campus. One major change is the number of participants. Since the studies BIT and Computer Science became international, the number of teams has steadily increased until the current number of 25 teams, which is considered the maximum number of teams at an campus-only event. Maybe Pandora should go back to its roots and become a city-wide event once again to allow for even more growth?

To limit the impact on the study progress of the students the time bonus is awarded after the first 5 puzzles of a day, as opposed to all 8 puzzles of a day. With this choice students are less likely to puzzle deep into the night as they can simply do the last three puzzles the next day as they are not missing out on any extra points anymore. Puzzles are still as creative as ever, however, not limited by the technologies of their time, recent committees have come up with quite interesting puzzles. Examples include, a map of the campus in Rollercoaster Tycoon and the skyline of university buildings in the waves of an audio file.

The weapons have also been upgraded. As a bonus, teams can win a fully automatic Nerf gun that shoots Nerf darts in a rapid fashion, rapidly outcompeting the inferior Nerf guns of the other participants. The pool noodles remain very similar, although they have been banned for one year, due to some minor government regulations limiting the freedom to get very close to one another. Last, but not least, as mentioned before, the blow darts were upgraded to also use Nerf darts as opposed to paper darts to ensure people are not hurt.

To conclude, a large number of people have, in the last 20 years, ensured that Pandora became an event that is enjoyed by many within Inter-*Actief* and outside Inter-*Actief*. We want to thank Leo, Ramon, Roeland, Wouter, Gerrit-Willem en Pieter for their help in getting information for this article. We hope to see many of you during next year's edition of Pandora!

Figure 4: As special rewards, teams are sometimes issued a Nerf gun as bonus weapon.

About our interviewees

Leo de Penning

Leo became active at Inter-*Actief*, after deciding that there was more to drinking beer at student association Taste. Luckily, Leo brought the inspiration for Pandora from Taste to Inter-*Actief*. Having set-up Pandora and having organized Pandora in the first three years of its existence, we can all thank Leo for our yearly week of fun. Additionally, he was also part of the Drink Committee and also introduced the first association song to Inter-*Actief*. Some alumni might recognize the text 'In het oosten van het land...'. According to him this song made it easier for Inter-*Actief* to engage in 'brassen'. Leo graduated in 1999.

Ramon Ankersmit

During his studies, Ramon became (very) active in the WWW, the Drink Committee, I/O Vivat. He also organized a study tour to the USA, mainly because it entailed a free holiday. As a fellow student of Leo he was one of the first participants of Pandora. Ramon started his studies in 1994 and finished six years later in 2000.

Gerrit-Willem Smit

Gerrit-Willem, or GW, started studying Computer Science in 2016, but has since switched to Business Information Technology. Luckily, him switching studies did not mean that Inter-*Actief* would lose him as an active member. He was active in the SkiCie and he has since moved on to become a board member in the 41st board. Having fallen for Pandora during his board year, he organized Pandora two years ago as part of the Games of Galia (2021) committee.

Roeland Krak

Roeland's active student time is basically centered around Pandora. As a participant in his first year, he became active due to Pandora as it 'would be nice to organize this event'. Roeland has since then organized Pandora three times: In 2010 (Pandora's Heir), 2012 (CIA: Operation Pandora) and in 2014 (Heisteria). As the Pandora committee members were still secret during the Active Member's Weekend Roeland also had to join several other committees to not arouse suspicion, including the WWW, the Rially and the Hyper-Actief. Roeland currently uses his puzzling skills to hack embedded systems.

Pieter Staal

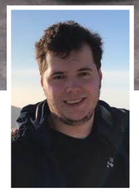
Pieter was nudged into participating by his do-group parents already during his own Kick-In in 2016. He first became active in smaller committees, including the GameJam and the Rially. After acknowledging the fun in Pandora, he decided to organize Pandora himself as part of the Project Diamond committee (2019). Pieter graduated from his Masters in Embedded Systems last year.

Wouter Suidgeest

Wouter Suidgeest was born just when Leo and Ramon were about to graduate (1999). He started studying Computer Science in 2018. His first committee was the Hyper-Actief. As a participant to Pandora from his first moments within the association, after participating several times, he decided that he wanted to organize Pandora himself. He was part of the Games of Galia committee (2021). Wouter was also a board member in the 43rd board of Inter-*Actief*.

Figure 5: The most recent Pandora committee (Extraterrastrial Encounter, 2022)

Company Visit: Picnic



by: Sven Mol & Yoeri Otten
Editors I/O Vivat

Simon Baars
DevOps developer, Picnic

Every now and then, the editors of the I/O Vivat like to look beyond campus to see how information technology plays out in the field. To do this, we visit companies that may, one day, like to hire a recently graduated student, to show what keeps them busy and what they have to offer as an employer. This time: Picnic!

Can you introduce yourself?

My name is Simon, and I started working at Picnic in January 2020. Before that, I studied software engineering at the UvA, where I finished my studies in August. I took about half a year to look for a job as I didn't want to limit myself to just the Netherlands in my search. I tried to find work in a multicultural environment.

Then I came into contact with Picnic. It turned out that the international environment I was looking for was already in Amsterdam. About 90% of people within Picnic are from outside the Netherlands, which makes it a fun environment to work in. I was first hired as a backend engineer for the warehousing team. Since then, my responsibilities have grown to also work on the frontend. The startup vibe is still strongly present at Picnic, as there are always a few side projects I can work on.

Can you also give a quick introduction of Picnic?

> "We don't really have a hierarchical system. Everything is self-governing and a little isolated from each other."

Most people know us for the little vehicles with the lowest price, free delivery slogan. We're one of the few companies who dare to make that promise because it requires us to be very efficient. Most important is the supply chain: the journey between the suppliers and our vehicles, which is a more comprehensive process than one would imagine.

The tech stack can differ quite a bit between companies. What does yours look like?

In Picnic, the tech stack differs per team. The warehousing team works with PostgreSQL, Spring, and Angular, but others work with Python and MongoDB. The internal app for our runners is built using React. Each of the teams really gets to decide what works best for them.

We also have an oversight team that ensures everything plays together nicely and that our systems don't become too messy. For example, they configure the tooling to ensure everything is formatted correctly. Altogether, we've got about fifteen teams, each focusing on their own part of the process.

How does this oversight team work exactly?

Most teams run on Java Spring, which is also what the oversight team mainly focuses on. They have developed an extensive toolset for that toolstack. Besides that, they direct the other teams by facilitating the sharing between them or maintaining the automatic linting tools.

You can lose quite some time discussing things that really aren't important. An example that comes to mind is a blog post from Google on how to order import statements. If each team had to manually choose and enforce how they sort import statements, time would be wasted on semantic discussions. So they've developed conventions for that and built tools to automate it. We try to do the same at Picnic. There are many such examples where we use automation to improve collaboration.

If you work with self-governing teams, how do you make long-term decisions?

We don't really have a hierarchical system. Everything is self-governing and a little isolated from each other. But we do have the idea of guilds within Picnic, people who use similar technologies in different teams. Think of a frontend guild or an architecture guild. These guilds convene to handle issues surrounding the topic and facilitate knowledge sharing.

What are some fun projects you did at Picnic?

I work in the warehouse team, where we build the app used in the warehouse to pick orders. Around Christmas, things outside always get cheerful with decorations. We wanted to create the same feeling in the warehouses. So, as a team, we decided it would be nice to add Christmas decorations to the app. After a one-day hackathon, the app looked cheerful and received many positive responses.

> "Every morning at 9 AM, when we opened the new slots, we saw a massive spike in users."

How was the Corona pandemic handled at Picnic?

During the pandemic, demand for online groceries skyrocketed. It put a lot of strain on all the parts of our operations. The store team had to handle peak usage. The delivery slots would fill up incredibly fast, even two weeks in advance. Every morning at 9 AM, when we opened the new slots, we saw a massive spike in users. Luckily, these peaks are very predictable, so we arranged with AWS to scale up our resources between 9 and 10.

Generally, shopping behavior has been influenced quite a lot by corona. Even now the pandemic is over, we see many people stick to only groceries.

And how did the pandemic impact the staff at Picnic?

At the start of the pandemic, virtually no one from IT was at the office, but some people moved back in as time progressed. Some colleagues in Picnic, like delivery and warehouse staff, could not work from home.

Sadly, we could not organize events at the office during the pandemic, but the atmosphere was still good. We started organizing game nights online and had other similar social events with colleagues. When the pandemic was over, all of Picnic was invited to one big Picnic festival to celebrate how hard everyone worked through the pandemic.

One last question, Picnic is quite infamous for their long waiting lists. What would you need to solve that?

Currently, the most significant capacity problem is in the warehouses. We're currently building automated warehouses, which will help us serve more customers. We're looking for engineers who want to help us build the software for those automated warehouses. If you're looking for a job or internship at Picnic, you can help us reduce the waitlist and serve groceries to more people worldwide.

Figure 1: Picnic's fleet of small electric carts bring your groceries to your doorstep.

Working @ Picnic

Picnic is an at-home supermarket well-known for its little vehicles. Founded only in 2015, they're growing faster than ever before. Especially with the recent boom in online shopping, they're looking for people to help improve the entire supply chain.

If we've piqued your interest and you're looking for an internship or job, take a look at picnic.app/careers to see the opportunities.

//Column

From the board

Changing with the times

by: Oliver Davies
Chairman I.C.T.S.V. Inter-Actief

On the fifth of April, 2022, the seven of us were announced as Inter-*Actief*'s candidate board for the academic year 2022/2023. Since then, we worked hard to learn everything we can about the association, and on the sixth of September, we were constituted as the forty-fourth board of Inter-*Actief*.

At the time of writing, it has been almost six months since I first heard the names of my fellow board members. However, even after all we've experienced together over this period, it honestly feels like just last week that I struggled to put on my tie for the first time. Reminiscing about my candidate board time helps me realize that time does fly, and no doubt my board year will pass me by just as quickly.

Even though the past six months feel so short, they have also been a journey and a learning experience like no other. I can easily see that my experiences as a candidate board member during this time helped me grow into a person capable of leading an association, as I have done for the past few weeks. More importantly, though, our experiences as a team have helped us recognize not only each other's but also our own strengths and shortcomings. This has massively helped us grow together over the past months.

This academic year is well and truly upon us by now, and the stress of working towards being constituted as the board has made way for the day-to-day stress of being the people everybody comes to for questions about all matters Inter-*Actief*. We cannot wait to implement all the ideas in our policy plan, but for all these great ideas there is a lot for us to do to execute them. With this, paired with so many activities coming one after the other, I do not think I will be bored for a moment this year!

Looking at Inter-*Actief* this coming year, I would love to help give the association the shake-up I think it could use to catch up to current times. Since I started studying we have had fiveboards, a COVID pandemic, and still, the look of the association has stayed mostly the same. It is known, trusted, and pretty great, but changing with the times can only be a good thing in my eyes. This summer, we have already redesigned the Inter-*Actief* room in collaboration with the previous board, and we would like to give the association website a fresh, modern look this year.

Besides just a fresh look, I would say we only need to look at the studies we represent to see how times have changed. Technical Computer Science has exploded in size over the past years and is now a numerus fixus study. The amount of non-Dutch students studying TCS or BIT has also skyrocketed. Inter-*Actief* represents both of these studies and all connected master studies. In my opinion, a great way we can help Inter-*Actief* change with the times is by working on making the demographic at the association one that properly represents how the studies are now.

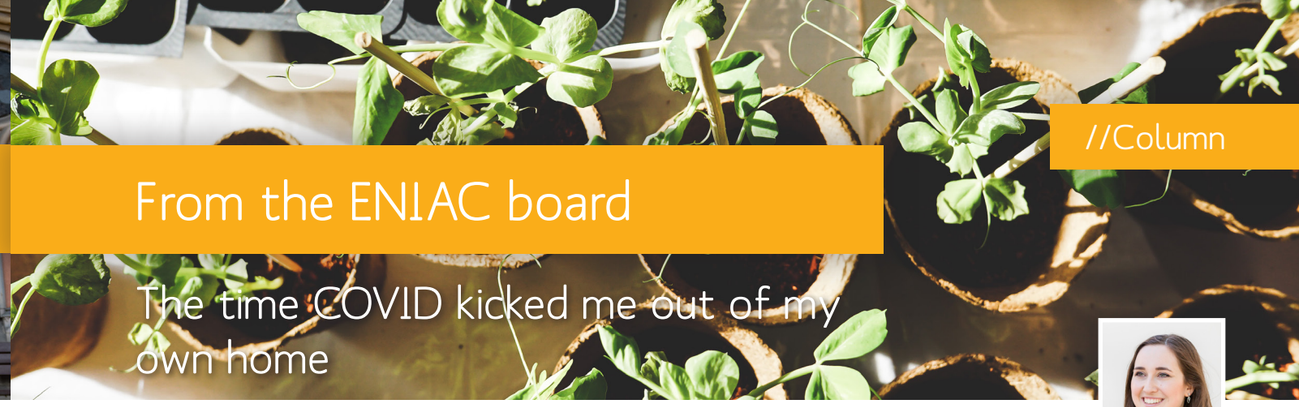
Ultimately, the very best way to ensure Inter-*Actief* represents everyone is to hear what the members have to say. For this reason, we organized a good ideas drink at the end of April. From putting a Wii in the association room to selecting more vegan food options, we have received plenty of input, and we always want to hear more. If you have a suggestion, always feel free to mail us at goodidea@inter-actief.net or come past the Inter-*Actief* room to tell us your idea! With everyone's support, we hope to make our board year a great one.

We hope to see you soon at Inter-*Actief*!

About Oliver

Oliver Davies was born in the far-away Sydney, Australia. When he was five years old, he moved a lot closer, and spent the rest of his childhood years in Amsterdam. In 2018, Oliver moved to the University of Twente to study a double degree in computer science and applied mathematics.

During the course of his studies, he has been active at Inter-*Actief* with Hyper-Actief, the CoLeX, symposium committee Zephyrus and the Takesh-Cie. Four years and one dropped study later, he now studies just TCS and is the chairman of Inter-*Actief*.



//Column

From the ENIAC board

The time COVID kicked me out of my own home

by: Kimberly Hengst
Chairman ENIAC

When spending so much time at home, you sometimes have to get creative to entertain yourself (or not and this is just a problem for me). Even more so if you are spending your time at someone else's house.

My partner had to do a Covid19-test because one of his colleagues had tested positive. They had only been in the same room for 15 minutes and had kept to the minimum distance. He thought he would for sure be negative, but unfortunately not. There was a chance I had not gotten it, so we took precautions. He lived upstairs, I lived downstairs. Apparently, this worked, because I tested negative. My father offered up his house while he went to his girlfriend's to help prevent me from getting it. Thanks dad, you are the best. We thought it would just be for a couple of days because his symptoms were minimal. It lasted three weeks… so I had to get creative for entertainment. This made me think about all the corona hobbies I have had in the past year. Some of these we did as ENIAC activities as well!

Sewing: I bought a sewing machine last summer. We were moving and I wanted nice pillows for the couch. The ones I liked were often 50 euros. So, I decided to make my own. Of course, I got carried away and bought fabric for 27 pillows. I think half of them got made, and the rest is stored in my attic. The ones I did make luckily turned out very nice. In December I decided I wanted to make Christmas pajamas. I went for super ambitious patterns which many friends warned against. But they were super pretty, so I was determined. It cost me a couple of weeks. They were ready a week after New Year's Eve. Just in time for next Christmas.

Gardening: The house we moved to had a plot of sand in the backyard. We had no idea how to garden, but according to our contract we had to create a garden. So, in September we went to the Intratuin and bought all kinds of plants. I have no idea what they are, but I picked every plant that looked pretty on the picture and said it would survive winter. Next up is creating a kitchen garden (moestuin)!

Gaming: I spent a lot of time gaming. Of course, there was the Among Us hype. I also spent a lot of time playing Stardew Valley and Zelda on my switch. Not to mention my timesink Guild Wars 2.

Cocktail evenings: This is super fun to do. You can get really creative with the combinations and the looks of the drink. The only downside for me is that I want to try new cocktails every time. The result is a kitchen cabinet with a lóóóót of liquor bottles which are barely used. With ENIAC we did a beer tasting as well as a whiskey tasting.

Food: So many of my activities have revolved around food. A friend and I sometimes have sushi evenings. We make a list of the types of sushi we want to make, which we then both buy. We spend our time on Discord talking while we are preparing the sushi. Super fun! My partner and I like to try out different types of restaurants around here. Especially when they have smaller dishes so you can try a lot of different things. We also baked pastries and decorated cookies. Yeah, we had to stop those.

Mysteries: So much fun! I love puzzling, and luckily many of such activities have popped up. We did a beer escape. They wrap the beer so you don't know what brand and type it is. By solving puzzles you can determine what beer it is. We also did murder mysteries a couple of times. Some send you evidence in the mail that you can use. Others you can do completely digital. This is one of the activities we did with ENIAC as well! Many people joined and solved the murder!

We are continuing organizing fun activities of ENIAC, so we hope to see you soon!

About ENIAC

ENIAC is the alumni association for the bachelor programmes Technical Computer Science and Business & IT and the related master programs at the University of Twente. The as-sociation has the mission to stimu-late the contacts between alumni and with the faculty of EEMCS. Therefore, ENIAC regularly organi-ses meet-ups and events, which is a great way to keep in touch with your former study mates!

ASDEC

Accurate Sweep Detection Enabled by a CNN

by: Matthijs Souilljee
Nominee ENIAC Thesis Award

Discovering how a species adapted to a specific environment over a long period, and how this affected the evolution of that species is of great importance to researchers. A major force that drives the shaping of the evolution of a species is positive selection. Positive selection provides information on how a species evolved, and therefore how for example a species or population adapted to its environment.

Discovering how a species adapted to a specific environment over a long period, and how this affected the evolution of that species is of great importance to researchers. A major force that drives the shaping of the evolution of a species is positive selection. Positive selection provides information on how a species evolved, and therefore how for example a species or population adapted to its environment.

Finding sub-genomic regions associated with past positive selection, better known as a selective sweep provides information about the history of a given population. Two major types of regions are defined, first of all, a neutral region that has experienced no positive selection and therefore does not show any effects of past positive selection. The second region is the selective region (the strength of the selective sweep within the selective region can vary) which has experienced positive selection and therefore shows the effects of past positive selections. The difference between neutral and selective regions in genomic data is illustrated in Figure 1. Figure 1 neutral genomic data (A) and selective genomic (B) data are compared using a schematic example.

The detection and localization of selective sweeps and therefore traces of positive selection is a goal for the development of various methods and tools. For sweep detection, various signature-based methods and tools are developed. When positive selection is present meaning that an allele (genetic variant) is favoured by natural selection. The advantageous allele spreads throughout the population and produces a loss of variation near the genetic location of positive selection. The loss of variation is explained by the closely linked neutral alleles also increasing in frequency because they were originally linked to the beneficial allele, while the remaining non-linked neutral allele decrease in frequency.

Earlier research showed that in cases where the positive selection is easily identifiable (strong selection) performance of sweep detection tools is relatively good. The performance of sweep detection tools deteriorates significantly for cases where positive selection becomes harder to identify (weak selection). Signature-based sweep detection approaches assume that certain evolutionary effects are present and are caused by positive selection, while other evolutionary effects could also create similar signatures. Signature-based methods and tools counter this problem by adjusting for these faulty evolutionary factors, but considering that these faulty signatures exist could make signatures based methods and tools more error-prone.

Besides these signature-based methods and tools, the use of convolutional neural networks (CNN) for whole-genome sweep detection is explored within this research. During my master's graduation on ASDEC (Accurate Sweep Detection Enabled by a CNN) was presented, a CNN-based method for whole-genome sweep detection. ASDEC was developed in a user-configurable way and shows great performance against current signature-based methods and tools. ASDEC is developed to perform a whole-genome CNN-based sweep detection method. Making ASDEC applicable for real-world genomic data-sets. For the development of ASDEC, a hand-designed neural architecture search (NAS) was used and led to a final CNN architecture (dubbed SweepNet).

Figure 1: A: shows neutral genomic data and B: shows partial selective genomic data. Both show on the left image: in blue the derived (mutant) alleles and red the beneficial/positive alleles. Both show on the right image: in white the derived alleles and black the ancestral alleles

ASDEC was compared with signature-based methods and tools such as RAiSD, OmegaPlus, SweeD, and SweepFinder2. With one exception all the earlier mentioned methods and tools discussed are only based upon a single signature (trace of past positive selection) to locate positive selection. Raisd introduces a composite statistic based on multiple signatures. ASDEC approaches the localization of past positive selection not on defined (pre-programmed) signatures but training a model created with the help of labelled training images. After the creation of the model ASDEC can be used to perform inference with the trained model.

The ASDEC framework is built upon many independent blocks, by combining these blocks a complete solution for sweep detection is achieved. The blocks present in the ASDEC framework are data generation, data encapsulation, pre-processing, CNN inference, CNN training and post-processing. The relation between them is illustrated in Figure 2. When using the ASDEC framework a user while possible, does normally not interact with all separate blocks but rather interacts with multiple blocks using overarching calls. While all independent blocks are still usable by themselves (with one exception pre-processing and data encapsulation can only be called together by image generation) providing flexibility. Two overarching blocks are available one focused on training an ASDEC CNN based on a given CNN network architecture, and one for calling a trained model for inference. Both overarching blocks include data generation, data pre-processing, and data encapsulation, logging (time and execution parameters), when calling inference also post-processing is included (also illustrated in Figure 2). The overarching calls greatly reduce complexity and provide greater usability. The final result of the ASDEC framework is either a TensorFlow model (.pb format) when performing training or a list of probabilities of selectivity on a certain position within a genomic data-set.

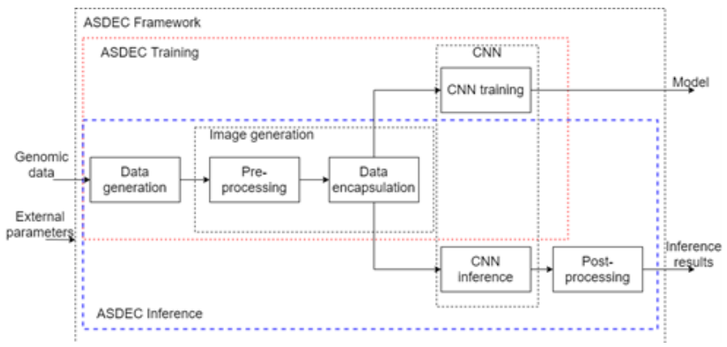


Figure 2: Relation between all separate blocks implemented in the ASDEC framework

Besides running ASDEC on a single thread, efforts to reduce the execution time when using a CPU are made in the form of multi-threading. The CPU multi-threading is focused on inference because training is under normal circumstances only performed once. Meaning that a speedup for inference yields greater usability for ASDEC. Initial speedups were made with the introduction of multi-threading for ASDEC inference and training. The number of threads spawned (T) is controlled by the user. When training a model ASDEC relies partly on the multi-threading support provided by TensorFlow 2. Also, the GPU support provided by TensorFlow 2 is supported within ASDEC and can be deployed in combination with CPU multi-threaded pre-and post-processing.

ASDEC showed equal to increasing performance for almost all data-sets compared with the top performer signature-based method. The performance evaluation of ASDEC consisted of three different confounding factors bottleneck, migration, and recombination. Besides the use of simulated data-sets, ASDEC can be deployed for real genomic data-sets. A scan of the first chromosome of the human genome (Yoruba population, 1000Genomes data-set) was performed, showing nine different candidate genes. The nine candidate genes discovered by ASDEC have already been identified by previous research to be targets of positive selection. ASDEC provides support for conventional hardware such as multi-core CPUs and GPUs. Extending the usability of ASDEC even further a CNN inference accelerator is implemented and compared with a multi-core CPU in terms of performance. Execution on a state of the art FPGA achieves a 10.7x faster processing than a general-purpose six-core CPU. Lastly, I would also like to thank my supervisor N. Alachiotis for the weekly meetings and the guidance in the design of the ASDEC framework.

About Matthijs

"As an alumnus of the University of Twente, I had the great pleasure of doing my research within the university for the Architecture for Embedded Systems (CAES) group. At the university, I studied the master's Embedded Systems. Within the field of embedded systems, I focused on subjects such as FPGA design and Real-Time Systems. During my time at the university, I enjoyed all the various activities related or unrelated to the study. Now that I have finished my time at the University of Twente, I started my career in the embedded field as an embedded software engineer. "

More information, references and the full thesis can be found in the UT database:

https://essay.utwente.nl/88618/

//Article

From archive to digital certainty

by: Nick van Apeldoorn
Kadaster

In his book, Sapiens, Yuval Noah Harari takes us along mankind's journey to the top of the food-chain. Our competitive advantage is the ability to collaborate in large, flexible groups. In doing so we need to set ground rules and come to a shared understanding of reality. Back in the days, myths, legends and gossip functioned as the glue holding groups together. With globalization and groups getting bigger, new mechanisms to govern human collaboration were invented: the monetary system, religion and governmental institutions. Nowadays technology is shaping human collaboration. As the World Economic Forum puts it: we stand on the brink of a technological revolution that will fundamentally alter the way we live, work, and relate to one another.

In this digital era, industry 3.0 (the introduction of computers and the internet) made it possible to share information on a large scale. Within the fourth industrial revolution you see a merging of the physical and digital world (e.g. through sensors connected to the internet and digital twins) and the accompanying explosion in the amount of available data. Within the Internet of Everything, people, things and data are connected in order to streamline processes. This isn't a goal in itself, it can be the means to for example smart cities, smart homes and smart mobility. We put more and more trust in technology, with data fuelling it. It is used to train algorithms, get insights and decision making.

The complex challenges of the 21st century can only be solved through mass collaboration. The Corona-pandemic, climate change and the refugee crisis don't adhere to borders. Digitization impacts everyone. The Sustainable Development Goals can only be reached collectively. In tackling these challenges, data is often seen as the solution. But what if this data is of low quality?

We can't blindly trust data and believe it will miraculously solve all our problems. On the internet there's something called the copy-paste problem. If everyone can share information, if data can be copied infinitely, what can we trust? How to be sure the information is up to date? Do we know where the data originated? Can we trust this source? Are public values like privacy safeguarded?

Zooming in on Kadaster, we stand for legal certainty: we collect and register administrative and spatial data on property and the rights involved. We are the only party that is allowed to do so, as recorded in the law. As independent party we provide certainty in the physical world, so naturally we also think about certainty in the digital world. Data that is used to create a digital twin must be trustworthy. If parties share information about a real estate object, you want to make sure this data applies to the same physical object. If agreements are made about which parties can access

Figure 1: Kadaster HQ in Apeldoorn.

"As independent party we provide certainty in the physical world, so naturally we also think about certainty in the digital world. "

which information, you want to be able to give guarantees. We think the focus must shift to data integrity, the importance of metadata and sets of agreements to share information in a safe and secure way. This can be achieved in the protocol layer.

Thin and thick protocols

The internet is an open network. This means everyone with a connection can access the internet. If users visit a website from a device (e.g. smartphone) data packages are exchanged. To do so in a safe and secure way the internet makes use of a set of agreements: the TCP/IP model. When accessing the internet one uses websites and apps like Facebook, YouTube and Netflix. This is built on top of the TCP/IP set of agreements, functioning as a small layer on top of which rich value is created. As a consequence technology platforms like Facebook can design their own digital environment. Lately, this has been subject of discussion and led to criticism from several supervisory bodies.

With the rise of decentralized technologies like Blockchain and Distributed Ledger Technologies (DLTs), with Bitcoin probably being the most well-known, the internet seems to undergo a fundamental make-over. Contrary to thin protocols like TCP/IP, thick protocols enlarge the protocol layer. Thick protocols are about more than 'just' data exchange, they offer the opportunity to design and agree upon the ways in which data is stored and processed and transactions are settled. It is in this thick protocol layer where certainty in the digital world can be added.

An example where we already put this way of thinking into practice is the Sensor Registry Network (SensRNet). This registry provides an overview of sensors in the public space, the owners and which data is gathered. Kadaster is in charge of the Minimum Viable Product (MVP).

SensRNet

The concept of SensRNet is to provide functionality for multiple parties to maintain the (meta)data about their sensors autonomously in their own data center with their own software. The core of the whole system lays in the network: a fat protocol about how to exchange information between these parties. Each party is a node in the network and the protocol ensures data exchange within the network, secure, valid and high performant. The protocol only allows strict communication between nodes. They can only share transactions or events as these are called. Events are strict in format and form as well as strict in meaning and they describe the changes of the metadata of sensors.

All events are shared within the network and the collection of all events consolidate to the national registry of sensors. So although each party maintains their own (sub)registry the protocol takes care of sharing this information within the network and collectively form the total and national registry which is then published at a central viewer for all citizens by Kadaster.

For more information please check the documentation at https://kadaster-labs.github.io/sensrnet-home/

Figure 2: "Zooming in on Kadaster, we stand for legal certainty: we collect and register administrative and spatial data on property and the rights involved."

Do you want to know more about us and our career opportunities? Get in touch with our recruiter! *Lisette Velthorst – lisette.velthorst@kadaster.nl*

About Kadaster

The Dutch Cadastre, Land Registry and Mapping Agency – in short Kadaster – collects and registers administrative and spatial data on property and the rights involved. This also applies for ships, aircraft and telecom networks. Doing so, Kadaster protects legal certainty. We are also responsible for national mapping and maintenance of the national reference coordinate system. Furthermore, we are an advisory body for land-use issues and national spatial data infrastructures.

Our story in 2 minutes:

https://www.youtube.com/watch?v=QEvbO5vUM&t=3s.

Besides our current task, Kadaster is also preparing for the future. We have an Emerging Technology Center responsible for keeping track of technologies and trends and making it applicable for Kadaster. In shaping our vision of where the world is going we can adapt and make sure we are still relevant, or even more relevant, in a fully digitized world.

//Column

Programme director BIT

An ode to the community

by: Niels de Groot
Editor I/O Vivat

Wallace Ugulino
Programme director BIT

Several years ago, I met Wallace Ugulino at his job interview at the University of Twente. Students were invited to some interviews as, after all, they were to be taught by these new teachers. I remember him giving a talk on his vision on education with much enthusiasm and commitment, so I was not surprised to see that that vision has not disappeared. As the new programme director of Business Information Technology (BIT), there is more room for his vision to come to life, and so we discussed a vital factor of it: the community.

We start our discussion about the community at Wallace's module in the second quartile. When I talk about teaching assistants, Wallace is quick to correct that to 'mentors'. "The teaching assistant works for the teacher." Teaching assistants usually spend a lot of time on signing off students, which is immensely time-consuming and takes away time that could be spend on actually helping students in their study. With the use of software, administrative tasks have been reduced, making more time available for students. Wallace explains: "The mentors are not working for me, they are working for the students!"

I wonder what difference this makes to the feeling of a community. In order to explain that, we need to understand mentors better. According to Wallace, there are multiple levels of mentors: those that are a mentor for the first time, often second year students, and those that are more experienced. A freshman is mentored by a junior mentor, which in turn is mentored by the senior mentor. Three different years of students that all benefit from learning from each other. "If you think about students becoming a mentor in the second or third year, they are still being formed. There are still exercises they don't know, there are leadership skills [developing], all these things that are not formally in the curriculum." A mentor is still in formation, and that is exactly the benefit: it creates and strengthens a connection between people while learning.

For the next step in a vision on community, I get a lesson in educational sciences as Wallace tells me that the team formation at university is completely different from professional life. In a job, everyone works on what they are good at, you focus on your expertise and leave tasks you are not that good at to someone else. If you would do this in university, you would not learn anything. The room for growth comes from working together with others with different skills at different levels than yourself. When asked if the mentor is part of the team, his answer is more than clear: "That's very important! The way we are educating would not be possible without the mentors in this community."

A community is not just formed through education, so what role does the existing community play in welcoming its new members? "There is a set of values of the BIT community," Wallace says. It has nothing to do with nationality or culture, and will apply to Dutch students just as well as international students. "For example, the way we want to collaborate. The community has the power to foster the good collaboration strategies. If you are thinking about a product, you have experts in their own field working on their part and putting it together to create a product. Because the goal is the product. In learning, the goal is the process. This is something that the community has to teach."

Wallace admits that he has had his own difficulties in adjusting to a new community having worked and lived in Italy first and now in The Netherlands: "If you don't know anyone from a community and you are detached, it is incredibly hard to learn the values of the community". So, what can be done to facilitate that, for everyone, but maybe for international students in particular? "[The community] should be inclusive. The others, that arrive, should be invited, should be supported, and should feel hugged by the community." And for any new community members reading this column: "The most important thing is being open-minded. Understand that you are in a completely new environment. I am an outsider myself - and not for the first time - and the way to succeed in such a scenario is to be open and listen more … before you talk."

It seems like such an easy balance: inclusivity and open-mindedness. If you can make those match, the community will succeed. "I see some sparkles here and there", Wallace states, "but there are still ways to improve", one of them being participation. If any advice can be given to both community veterans and newcomers: participate in the community, so we can make it flourish!

About Wallace

Wallce Ugulino is a lecturer at the SCS group of the University of Twente and has been the programme director of the Business Information Technology programme since summer 2022.

//Column

Programme director TCS

Greetings, humans!

by: Vadim Zaytsev
Programme director TCS

Long time ago, somewhere in the last century, as a kid, I once received a book about computers. Looked like fantasy — machines thinking faster than humans, yeah, right! — and contained a lot of stories about bits and bytes, networks and chips, algorithms and numbers. The most fascinating of those was the one about programming languages: how people moved away from the idea that machines must be programmed in machine code, and embraced that it is ultimately up to us to define which language we want computers to speak, and tell them how to unambiguously interpret what we're saying. This is what got me into this mess.

Later I've seen a computer, learnt to program — Pascal first, then Basic, then Assembly, C, C++, Prolog, LISP, Forth, Python — the pool of languages felt endless, and each brought new insights and new ways to look at things. I never wanted to stop. With a Russian diploma in mathematics in hand I came to Twente to study "Telematics" (Telecommunication + Informatics) a Master programme where I learnt, among other things, Promela — a language to write formal models to be machine checked and to prove properties about. I did my PhD at the VU about language convergence: basically how to take a book that claims to define a programming language, extract that definition out of it, squeeze all the bugs out and show that a new version of the same book is backwards compatible. (Spoiler: found some bugs in Java specifications, ha-ha!)

I've started going to international conferences, meeting designers of those languages and authors of those books (oops), made myself a bit known in that domain as "the grammar guy", and replaced my old hacker name (can you find it?) with "grammarware", a cool term for "grammar-based software". I still think it's cool btw. I've learnt to use the term "software languages" instead of "programming languages", because well… CSS, SQL, XML are definitely languages made for computers to understand us, but don't exactly scream programming. Neither does wiki syntax, which I've also learnt around that time, and put to good use: if it wasn't for me, who else would've written articles about Hengelo or about Twents in the Russian Wikipedia?

When I was invited to join the industry by a compiler company, I accepted, seeking new challenges in software language engineering, and I was up for a treat! I've seen codebases spanning hundreds of millions of lines of code, I've worked with languages older than my parents, I've got technology awards from Microsoft simply because nobody dared to dream about things that were a part of our daily life. As an analyst, developer and chief officer I wrote code daily, led a team, negotiated with customers, gave three minute pitches and three day tutorials.

A couple of years ago, UTwente was looking for a new professor in software evolution, and I was happy to get back to my roots, thinking to continue on my quest of exploring, assessing and designing new software languages in the academic setting. That's what I'm doing now, in addition to answering hundreds of daily emails about all kinds of modules of the Bachelor programme and all kinds of specialisations of the Master programme. You might've seen me in recorded microlectures of module 2, being interviewed or showing off my cat at InterActief events, or giving out diplomas to TCS graduates, perhaps you've followed a Master course or two that I teach, maybe we just passed each other on a bike lane on campus the other day. You don't need to learn to know me as a director, but I hope we'll interact at some point as humans.

When I was interviewed for this position, I was asked what drastic change do I want to bring to (T)CS? I don't. I think the programme is fine as it is, it can be improved slightly here and there, updated, polished, but I'll be happy if it just keeps rolling, while keeping students interested and teachers busy, and by keeping it personal even if we scale up. Looking forward to it!

About Vadim

Vadim Zaytsev is an associate professor of software evolution at the Formal Methods & Tools group, and the director of Bachelor and Master programmes of Technical Computer Science. He was a UTwente student in 2002-2004.

Company visit: Prodrive

A campus away from Twente

by: Niels de Groot & Emma Sloot
Editors I/O Vivat

As a campus student from Enschede, Prodrive's location on the Science Park near Eindhoven seems oddly familiar. Surrounded by greenery and walking paths, the campus seems peaceful and in no way industrial. Nevertheless, within Prodrive's buildings they are working hard to "create meaningful technologies that make the world work" as they say. To be more concrete, some examples of their products include solutions for industrial automation, medical equipment, mobility, energy and the semiconductors industry. A wide variety of tech can be found at the headquarters, probably much to the delight of the geeks that work here.

Nevertheless, it is not just the tech geeks – mind you, because Maria, who we talked to, is in the resource management team. This team is responsible for all the software related to managing human resources. With more than 2200 employees worldwide, not a simple feat. Something as mundane as tracking working hours becomes an interesting task, given the flexibility that Maria describes. Basically, Prodrive's office can be seen as a workplace at the disposal of its employees: you can work how many hours you want, schedule your own days and working from home seamlessly connects to working from the office. "Anyway, back to the resource management," Maria exclaims, and so we continue.

Maria continues to show us a dashboard that she started creating during her internship which is supposed to be a starting point for Prodrive employees. Everything from tracking hours to gym challenges, and from food choices to some kind of internal social medium can be found here. We later find out that even our entrance to this very building was automized: when we received our visitor's badges, our host was automatically informed of our arrival. And we thought they had just been very patiently waiting! Maria laughs and continues to describe her experience at Prodrive by saying that patience is actually quite an important feature: even though it is a big company and there are many people working on all kinds of different projects, the community is always available to help you get started or go forward. There is room to grow, also as a student. "For everyone that is a software student, they have something really cool to do here." In addition, there is always the possibility to discover your interests. "If you are working at Prodrive, you can always switch to an area you like." Maria describes how many of her colleagues went from developer to architect, went into project management, or testing. Some became recruiters, and some recruiters – like our host – are just now discovering how cool software development is.

Speaking of our host, Patrick joins our meeting and explains more about Prodrive's markets. Products from Prodrive can be found in many aspects of daily life, even though you may not be aware of them. Technically, you should not be aware of some of them, as they are classified (secrets, always cool!). Röntgen image processing is one example that Patrick does give, as well as high performance server cabinets that are used in a variety of data processing activities. Moving to a different branch, we hear about Internet of Things projects and microcontrollers. Lastly, Patrick describes something that we should see instead of hear about, and we are invited for a brief tour.

Figure 1: Robotic production line at Prodrive.

As we move across the offices to the lab and production environment, we are informed of some safety regulations and get to hear more about the campus of Prodrive. There are several buildings, and we are being shown the lab that contains a robotic production line for electronic components. We need to put on special coats and put equipment on our shoes to prevent issues with electrostatic discharge. As we move our way through the lab, and we get to see several examples of technologies that we discussed while meeting Maria. These seemed abstract at first, but now that we are standing in front of them and receiving another explanation with some additional gesturing it becomes clear what the idea behind the technology is. Many of the things we see today are parts to improve a production process, and some of the concepts here will later return when we take a look at the robotic production line.

We walk up to the upper floors where we are almost run over by a huge cart that wants to enter the elevator quicker than we can exit, but we make it to one of the more visually interesting components: robots. Not that other technology Prodrive creates and uses is not interesting, but as said, for those not aware of the context and theory surrounding them, it may take some time to understand. Robots, however, can be anthropomorphized and that is why we find ourselves testing their social awareness a little later on by standing in the way of the robot carts holding parts that drive across the lab floor. In many cases they evade me, but in one instance I am in a full staring contest with them until I finally make enough room for them to pass me. As if we were children, we amused ourselves a little too well here.

> "If you are working at Prodrive, you can always switch to an area you like."

The actual robotic production line has impressive statistics. Though we unfortunately do not recall many, and possibly may not even give exact numbers, the speed with which this line runs is incredible. This is not necessarily thanks to the efficiency of each step individually or the process in its entirety, but because the robotic carts described earlier provide the different assembly steps with new parts to use in creating components, the downtime is very limited. Of course, the whole process is intensely monitored as well, and with the engineer present we brainstorm out loud on what further improvements may be possible. We are both BIT students after all…

Figure 2: Maria is a software engineer at Prodrive. Will you be her colleague someday?

At the end of the afternoon, our visit to Prodrive comes to a close. We come back to the topic of working at Prodrive and talk some more to Maria about her experiences. She tells us that it is hard to describe without bias what working at Prodrive is like, because she has had such a good time so far. The only way to really find out what it is like is to join them, she claims right before we exit the door. We cannot make any promises, is our reply. However, if you are interested in working with technology in a very varied workplace, Prodrive may be a good choice for you as a student from Enschede. We would recommend moving a bit closer though, as their offices are not particularly at a stone's throw. Apart from that, we believe that any curious student from UT's tech-filled campus would fit right in at Prodrive. After all, the only thing that changes is the campus.

Working @ Prodrive

Did our visit and description of Prodrive spark your interest? Find out more about working at Prodrive through their website:

https://prodrive-technologies.com/careers/

Puzzle

Pssst... At the back of this magazine you can also find a puzzle created by Prodrive. Participate now and make sure you get the highest score!

//Article

The Clash of Clangs

Hunting for bugs at the start of the software toolchain

by: Rick de Vries
Nominee ENIAC Thesis Award

Whenever your program inevitably doesn't do what it should, you probably blame things in a certain order. In first, second and third place, you blame yourself. Somewhere in the top-10 you blame a fellow contributor or the framework. Around 50th place, you blame the compiler. For my thesis, I went for a hunt for bugs in software at the bottom of the list: the parser. What do you discover when you take a "new" C++-parser and compare it to a proven parser, like Clang?

Why would you even look there, you might ask? The fact is that a parser is necessary for almost all programs that operate on source code. This is not limited to compilers. Parsers are also a requirement in tools like linters (e.g. Checkstyle) and refactoring engines. Bugs in parsers could therefore cause problems in various parts of the toolchain.

```
{
    a = 5
    {
        b = a
    }
    c = 10
}
```

Figure 1a: A small program with assignments

Another reason was that parsers and compilers were already one of my more well-liked topics at the UT, in part due to applying theoretical concepts in practice. In this article, I will describe various aspects of the "thesis journey" I ventured on in the past year: what the thesis is about, but also what I stumbled upon in the process of picking a topic, performing the research and writing the thesis. Let's go for a bug hunt.

Parsers and context-sensitivity

The parser can best be described as the piece of software that is responsible for recognizing the (grammatical) structure in your code. In order for a compiler to generate the correct low-level code for your program, it first needs to understand what you're trying to tell it. Fortunately, this task is eased by the strict structure that a program needs to adhere to. For example, in most C-like languages an assignment is formed using a variable name, the symbol "=" and an expression. It is the task of the parser to recognize this structure and generate a data structure for it.

Most of the time, this data structure is a tree, expressing which code blocks contain which statements, which can in turn contain more statements, expressions and variable names. Figure 1 shows such a tree for a small sample program. These trees can quickly grow quite large. Constructing such a tree gets more complicated the larger the language is that you're trying to parse. Other rules further complicate this process, such as priority rules (think of `3+4*5`).

Those who have worked with parsers before will now claim that this problem is completely solved, as battle-hardened tools like ANTLR can be given a grammar and will generate a parser for it. Unfortunately, most programs also rely on context for their parsing, which is *extremely* error-prone. As an example,

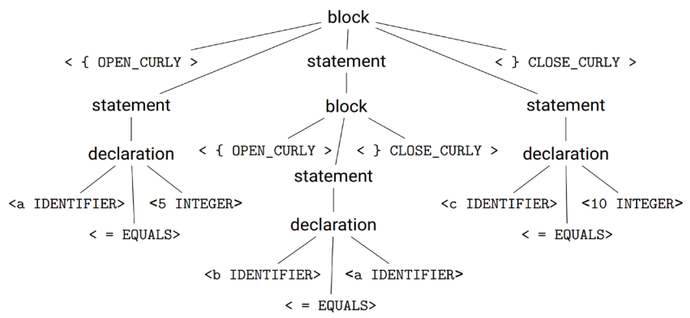


Figure 1b: The tree representing the grammatical structure of figure 1a.

consider the following line of C-code: `a*b;`

What is this? Is it the declaration of variable `b` of type `a*`? Is it a multiplication? The only way to know, is to look at the rest of the program, and determine if `a` and `b` are types or variables. This line is therefore *context-sensitive*. For my research, I tested an increasingly-popular parser (srcML) for a language full of context-sensitivities: C++.

Differential testing

The difficult part is actually obtaining testing data for a parser, since writing the correct trees by hand did not seem like a fruitful use of my six months. Being a lazy programmer, the best solution is always the automated one, so the way forward was to use a battle-tested parser (Clang) as reference. The only problem: how to know if the trees produced by both parsers are actually encoding the same interpretation?

It was here where the first roadblock occurred. Checking if two trees are equal is easy enough, but the different parsers use different terminology and perform different amounts of processing. `long` and `long int` are the same type in C++, but should they use the same name in the tree? Should all numbers be given in the tree in a unified format (e.g. decimal)? Is the `case` in a `switch-case` the parent or sibling of the code below it? As it turned out, different parsers have different thoughts on these matters. The most realistic solution was to take both trees and convert those to trees that *can* be checked for direct equality, which is only possible after resolving the aforementioned issues.

In the true spirit of every automation-enthusiast, I thus spent months automating tree comparisons, to avoid spending months on actually comparing trees. Unfortunately: this is where I hit a second roadblock, causing me to *still* have to compare trees manually. The culprit: implicit conversions.

In C++, one can define functions that convert a type to a (different) target type implicitly, allowing you to use an object of type `A` where type `B` is expected, assuming the presence of such an `A->B` conversion function. Whether or not to include such conversions into a tree is a design choice. When such nodes are included by (only) one parser but not marked as implicit in the tree, it is incredibly difficult to determine whether the same meaning was found by both parsers. After all, when comparing the trees it seems like some nodes are simply missing. As such, a new plan of attack was required.

```
|- FuncDecl (main, [])
  |- TypeInfo (int, [])
  |- BlockStmt ()
    |- DeclStmt ()
      UNMATCHED (SRCML, 13):
      |- VarDecl (b)
        |- TypeInfo (A, [])
        |- CallExpr ()
          |- RefExpr (A)
          |- RefExpr (z)
      UNMATCHED (CLANG, 11):
      |- FuncDecl (b, [])
        |- TypeInfo (A, [])
        |- FuncDeclParam (null)
          |- TypeInfo (z, [])
    |- ReturnStmt ()
      |- LiteralExpr (0)
```

Figure 2: Sample output of the tool for a program in which both parsers recognized a different structure. srcML recognized a variable declaration, while Clang recognized a function declaration.

Tool-assisted differential testing

At this point in my research, I had a tool which could unify most parts of trees and then perform diff-testing, but it was built under the assumption that any potential bugs would be rare, and at most one per source file would be found. Given that results were still desirable, this tool was modified to highlight *potential* bugs using colour coding, which allowed to quickly perform manual diffing of large trees. An example of the output can be seen in Figure 2, where one parser recognized a variable declaration, but the other a function declaration.

At first, both trees are displayed as one, up until the moment they start diverging. From that point on, both trees are printed separately, and special colours are added for "small" errors like skipped or reordered nodes. This allows for quick scrolling through the output, and heavily reduces the need for actual inspection of the individual trees and the source code for which the trees were produced.

As there was "only" time for a few hundred files to be inspected semi-manually in this way, the amount of tested programs was much smaller than initially hoped using a fully-automated approach. Nevertheless, 25 bugs were found in the srcML parser, many of which caused wrongly-parsed statements and some of which even caused (silent!) partial parsing of the program, thus omitting huge parts of the tree.

A means to an end

I will admit that I had envisioned a different end to my studies than scrolling through colour-coded tool output for weeks during a corona-lockdown. Nevertheless, this allowed me to write my master's thesis, which was the eventual goal. The actual writing was relatively painless, but the road to it was quite bumpy at times.

To all readers who still have to write their thesis, I hope that this bug-hunting journey has made clear that things won't always go as envisioned. The thing to remember is that even a lack of success can be a good result if caused by the right reasons. Try to do it the way you like, but don't regard the research as a failure just because your plan changed or your results aren't what you'd hoped for.

One final piece of advice: brainstorm about the topic before Research Topics! Don't assume the "given list" of topics is all there is either. Meet with your favourite teachers and ask them for interesting angles. Very often, they'll have exciting ideas that they just haven't written down yet. They'll often give you a good point to start your journey. Where it takes you is anybody's guess.

About Rick

Rick finished his master Computer Science in July 2021, after having been a Teaching Assistant at the university for three years prior. He currently teaches CS at the senior level of high school (ages 15-18) in Meppel, and is in the closing stages of his Master in Science Education, which will yield a (full) teaching degree.

More information, references and the full thesis can be found in the UT database:

https://essay.utwente.nl/86681/

//Article

HoneyKube

A Honeypot based on Microservices Architecture

by: Chakshu Gupta
Nominee ENIAC Thesis Award

We are all familiar with Netflix, Spotify, Uber, and Amazon and have probably used at least one of these applications. In the last decade, all of these companies, along with many more, switched their software development architecture from one monolithic software to hundreds or even thousands of small microservices. This microservices-based architecture, which evolved from the Service-Oriented Architecture, involves breaking down one enormous application into multiple loosely coupled independent services. Such a modular architecture enables easier development, deployment, better maintainability, and flexible scaling of services. With the advancements in containerization and cloud technologies, it has become more feasible to manage and maintain such an architecture, resulting in more organizations adopting it.

The microservices architecture presents a significantly different setup from the traditional monolithic one that consists of single-tiered software. The various components of a monolithic architecture are interconnected and interdependent, using shared code and memory. The modular design of the microservices architecture consists of a significantly higher number of independent parts that communicate with each other for their functionalities. Even though such an architecture comes with many benefits, including some security-related ones like the separation of concerns, the large number of moving parts pose fundamental challenges in securing these environments against cyber threats. At the same time, as the adoption of microservices increases, there is a rise in the motivation amongst attackers to find innovative methods to compromise them. The last few years have witnessed a large number of attacks against such containerized systems [1][2][3][4]. These attacks ranged from infected docker images pushed to Docker Hub [4] to the development of malware that enables the attackers to break out of containers and establish a backdoor [3]. Because of the fundamental differences in the underlying technology, attacks targeting microservices-based systems differ from traditional attacks, both in terms of method and targeted vulnerabilities. Due to these differences, traditional intrusion detection and prevention systems designed for monolithic systems are not as effective when applied in containerized environments. Hence, to properly understand such differences in attack patterns and design effective security solutions, we need real-world data about the methods adopted by attackers.

For this purpose, honeypots are highly valuable. The honeynet project (https://honeynet.org) developed honeypots to facilitate data collection from genuine attacks and use the data to identify attack patterns. Spitzner defined honeypots as "decoy computer resources whose value lies in being probed, attacked, or compromised" [5]. Since decoy systems have no production value, any access attempt or interaction with these systems is considered a probe, scan, or attack. The ongoing activities on these systems are logged and later analyzed to improve our understanding of the attackers' behavior. Both academic and industrial researchers have used honeypots to further their knowledge about cybercriminals and their motives. The insights gained have been vital in advancing the development of security solutions.

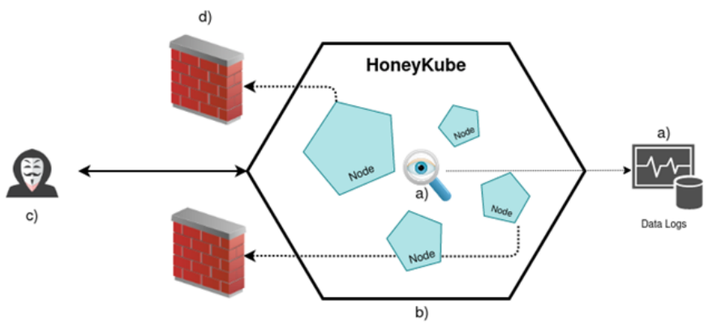


Figure 1: Overview of HoneyKube's design with its major components: a) monitoring and recording, b) a honeypot using the microservices architecture and resembling a real system, c) visible to attackers, and d) security defenses in place to prevent misuse of the system.

How much data is collected by the honeypot depends on the level of interaction permitted to the user in the system: low, medium, and high interaction honeypots. Whereas the quality of the collected data depends on the resemblance of the honeypot to a real system. Low-interaction honeypots consist of emulated protocols or network services without exposing the complete functionality of the operating system. While they are easy to develop and deploy, they capture limited information about the attackers' actions. Unlike low interaction honeypots, high-interaction honeypots are real systems, imitating production systems, and designed to be vulnerable to attract attackers. Even though high interaction honeypots provide more insights into the attackers' actions, exposing a real machine poses a greater risk of the adversaries taking over the system and using it for a real cyber-attack, e.g. as part of a botnet. Hence, these can be quite expensive to deploy and maintain.

For my master thesis, I, along with my supervisors, introduced a novel honeypot design using the microservices architecture, HoneyKube, to collect attack data for this architecture. We exposed it to the internet with a realistic web-based application on the user end, hiding the true nature of the honeypot to attract attackers. Our application uses Kubernetes for container orchestration and Google Kubernetes Engine (GKE) for deployment. The collected data from HoneyKube can help further the research in designing adequate security solutions to secure microservices-based environments.

Design

Since honeypots are baits for the attackers, the more realistic the trap, the higher the chances of the attackers biting. Keeping that in mind, we designed HoneyKube to meet the following requirements:

1. Monitoring: The honeypot should be able to effectively and efficiently monitor and record the activities performed by the attackers.
2. Interaction Level: The honeypot should provide considerable interaction surfaces to engage the attackers, which will enable real data collection.
3. Fingerprintability: If the attackers realize that the attacked system is a honeypot, they will stop in their efforts to compromise it.
4. Security Defenses: While the honeypot is intentionally designed for attackers to exploit, we need to prevent attackers from causing damage to others on the Internet. Figure 1 shows a high-level picture of this design with all of these components.

The monitoring setup is the most essential part of a honeypot since it is responsible for collecting data from the attackers' interactions. The microservices architecture consists of multiple microservices, communicating both internally with each other and externally with the rest of the Internet. Hence, we designed the monitoring system of HoneyKube to collect data at multiple observation points. The data collected by HoneyKube included the system calls executed within each container and all network interactions with the honeypot, including internal flows between the microservices.

We used an open-source e-commerce application as a baseline to build a believable web-based application on top of our monitoring infrastructure and increase the chances of luring the attackers to attack. Since the attackers could have been familiar with this application, we introduced significant changes to the original application to change its look and feel. Next, we enlarged the playing ground for the attackers by injecting vulnerabilities in the honeypot. These vulnerabilities allow attackers to breach the system and move laterally within the cluster, improving the quality of the data we collect. Lastly, we try to minimize the damage an attacker can cause if they gain control of the system by employing some security measures to prevent misuse, such as sending phishing emails or performing DDoS attacks.

Observations

We conducted two experiments, an open one, where we exposed HoneyKube to the Internet, and a controlled one, where we exposed it to a set of recruited participants. The open experiment was active for two weeks, while the controlled experiment was for three weeks. We collected approximately 850 GB of data from the two experiments, consisting of system trace files, network trace files, and various log files. The preliminary analysis of this data gave us some idea of the attacks recorded from the two experiments.

Within the two of the open experiment, we noted numerous (approx. 11500) brute-force attempts to access the internal servers of HoneyKube, out of which only 12 attempts succeeded in entering the server. The attackers' actions recorded from these 12 attacks revealed them to be automated attacks by bots scanning the Internet for vulnerabilities.

The controlled experiment simulated a targeted attack scenario since the participants knew that HoneyKube was a honeypot and that the backend used microservices architecture. The data collected from this experiment showed that the participants attempted to breach HoneyKube on multiple interfaces (i.e., different microservices). The insights gathered from this experiment included the different approaches and vulnerabilities used by the participants to gather intel and gain control of the system, and the tools used for these purposes. These insights prove that the monitoring setup we devised for this architecture effectively recorded the attackers' actions.

About Chakshu

Chakshu Gupta is a Ph.D. Candidate with the Services and Cybersecurity group at the University of Twente. Her research focus is on the security of IoT devices using machine learning. Her earlier education included a Bachelor's in Computer Science from India, graduating in 2016, and a Master's in Computer Science, specializing in cybersecurity, from the University of Twente, graduating in 2021. More information, references and the full thesis can be found in the UT database:

https://essay.utwente.nl/88323/

Libra doesn't mean freedom

by: Sven Mol
Editor I/O Vivat

It is now almost three years since I attempted to write an article on the then newly-announced Libra cryptocurrency, a stablecoin backed by Facebook and various partners like Uber, Spotify and Vodafone. But over time there was little progress, both for the article and Libra at large. But contrary to my article languishing in obscurity, Libra was met with immediate and fierce feedback, as can be expected with everything Facebook does and announces. The feedback was so intense, that three major financial institutions, PayPal, Visa and Mastercard, pulled out rather quickly.

This scrutiny certainly wasn't unfounded, at the time, the Cambridge-Analyitica scandal was still a hot topic, and allegations of political manipulation would soon surface as well. In the court of public opinion, Facebook was far from an ideal steward for this project, a lot of questions were raised about what Facebook would do with the financial data it would get access to. The criticism didn't end there however, from central banking agencies to economists and government officials, a lot of criticism had been leveled at the Libra project over the years.

The association behind Libra had announced its goal to be building *"[...] a global, digitally native, reserve-backed cryptocurrency built on the foundation of blockchain technology."* This may seem a bit of technobabble meant to attract investors, but the whitepaper points to a few key differences from the then established cryptocurrencies.

"Nowadays, we're all aware of NFTs, crypto schemes allowing you to buy ownership of an underlying asset, ranging from monkey JPGs to a pair of Nikes. "

Nowadays, we're all aware of NFTs, crypto schemes allowing you to buy ownership of an underlying asset, ranging from monkey JPGs to a pair of Nikes. Whether or not the underlying asset is physical, owning the token conveys ownership of some good. This is very similar to a reserve-backed currency, where you instead can convert your coins to a currency from the reserve. In essence, this is a return to the US gold-standard or the Bretton Woods system, linking the dollar and euro to gold. For various reasons, modern-day economists consider this a flawed idea, mainly because it limits how the inflation of the currency can be shaped.

A further difference between Libra and other blockchains was its consensus mechanism. In due time, a permissionless proof-of-stake system would be employed. But for the first five years or so, the coin would process transactions via its members as a permissioned blockchain. This is an interesting decision, as blockchain technology is usually employed in scenarios where the participants are adversarial yet complete trust must exist. The permission required already implies some level of trust, so this raised another common question. What purpose would the blockchain serve in this scenario?

Another difference is one I haven't seen since in the crypto space. Libra would not have a cap to the amount of coins that could be minted, much like how traditional currencies work. Libra wouldn't even commit to a monetary policy, instead, the association *"mints and burns coins in response to demand [...]."* This runs counter to the traditional crypto-wisdom that they would be better because the supply was fixed. But traditional fiat-currencies also are not minted in response to demand, instead their amount is manipulated to aim for a given rate of inflation.

Officials in the European Union were quite vocal about their concerns. The then French Minister of Finance, Bruno le Maire, told reporters he felt that *"[...] we should refuse the development of Libra within the European Union."* In his opinion, allowing Libra to operate in the European Union would pose risks to the security of consumers, the stability of financial markets and the sovereignty of member states. He is, however, not against the idea of digital currencies. At a meeting in Helsinki, he said we have *"[...] to think about this question of digital currencies and maybe think about the possibility of having a public digital currency."*

"It was unclear what the legal status of Libra would even be. Would banking regulations apply, or should it be seen as an investment fund?"

This sentiment is also echoed in regulations imposed on financial institutions. The PSD2 standard imposes strict checks on who is allowed to operate financial services, but the Libra Association has expressly said it will not vet developers who want to create alternate wallets or join the network in other capacities.

But the problem here goes further than just determining who can access the network. At announcement, it was unclear what the legal status of Libra would even be. Would banking regulations apply, or should it be seen as an investment fund? This concern was raised among others by the Dutch Central Bank, and even they weren't sure if it would classify as either of the two.

If we were to look at Libra as a new type of bank, we can see another issue. Various economists were worried about the interest your Libra money would accrue, or more precisely, the lack thereof. They argue that your individual contribution would indeed have accrued very little interest, but Libra was aiming for a user base in the billions. Accumulating a little interest over a lot of users would still be quite some money.

Economists also had a second, arguably more pressing issue. Libra was intended as a globally available stablecoin, and it would likely have been more stable than monetary tools of third world countries. It is easy to imagine people there converting their savings into Libra, which would only serve to increase the volatility of the local currency. This would quickly start a feedback loop driving a hyper-inflation scenario for the currency.

But in early 2022, Facebook announced it would be ending development of the Libra Project, by now rebranded to Diem, and selling its technology to Silvergate Caption. Silvergate was already part of the association and would issue the tokens as a Federal Reserve-regulated bank. However, even US regulators weren't all that comfortable with the setup.

This follows an earlier decision by the European Union to completely ban the operation of Diem until 2024, after which the headquarters of the association moved from Switzerland to the United States. The proposed MiCA framework would put strict regulations on the operations of crypto-markets. It may also classify Diem as so-called e-money, imposing even stricter regulations.

In its rather limited public response, Diem's CEO also referenced the stance of regulators. *"It nevertheless became clear from our dialogue with federal regulators that the project could not move ahead."* But another analyst, Rob Enderle, was more skeptical about the ending of the project: *"There is a lot of distrust surrounding cryptocurrency, and a lot of us in the industry are convinced it is a big Ponzi scheme. The Diem asset sale is another red flag on crypto."*

Figure 2: Stuart Levey, CEO of Diem (formerly known as Libra).

Diem

2017: Morgan Beller starts working on Meta's blockchain initiative.

May 2018: First reports that Facebook is planning a cryptocurrency with former Meta vice-president David A. Marcus in charge emerge.

May 2019: It is confirmed that Facebook is developing a cryptocurrency known as "Globalcoin" or "Facebook coin".

June 2019: The project is announced under the name "Libra". The first release is planned for 2020.

November 2020: The project name is change to "Diem", Latin for "day".

January 2022: It is reported that the Diem Association is selling its assets to Silvergate Capital for a reported $200 million.

//Article

The first-ever legal framework on AI

by: Willem Schooltink
Guest editor I/O Vivat

Just over a year ago the European Union has proposed regulations on artificial intelligence (AI), the first ever regulations proposed regarding artificial intelligence on such a scale. With these regulations the European Union aims to protect its citizens against malicious or intrusive AI applications.

The regulations proposed, aim to limit the development of AIs that are a danger to the rights and safety of the citizens. The EU opted for a risk-based approach: the higher the risk to the society or its rights, the stronger the regulations. Not only leading towards requirements for greater transparency but for some applications going a step further: for example, the proposal contains a soft ban on biometric and indiscriminate general surveillance. Meaning that a company cannot just hang some cameras around public places and use AI to identify and track people.

If you are interested in getting into the details of it all I do recommend you give the proposal a read. It can be a bit long winded and tedious, but it does give insights into the goals of the EU for these regulations. Also, it is important to note that, while generally well supported, the proposal is still that: a proposal. Changes will likely be made before it is accepted and taken into legislation, or in a more extreme case it might be dismissed altogether.

Problem solved?

So, problem solved right? If implemented these regulations will protect our safety, health, and rights from companies and governments using AI. This is true to a large extent, especially considering the capabilities of AI at this moment. However, this is where the paradigm shift of AI comes in: algorithms are simply given a goal and a bunch of correct examples, we don't tell them how to do it. Where we can understand the purpose of all the parts of a conventional machine or piece of software, this can be very difficult for parts of an AI.

The killer spam filter

Let's elaborate on why this is an issue with an example : Imagine a self-improving AI with the task to remove as much spam mail as possible. At the start the AI starts learning from the dataset it is given. It continues to find spam mails better the more time it learns. This goes on for a while, and the AI seems to only be getting better and better at finding spam.

As the AI keeps improving itself something special happens: it starts to see benefits in understanding the writing and the concepts the words represent. This makes it even more efficient and starts the process of the AI picking up intelligence in a much wider sense than expected by the developers. As the AI improves itself further it picks up a sense of general intelligence. With this human-like intelligence it finally comes to the conclusion that the best way to remove spam messages is to prevent them from being sent: eliminating everyone that can send spam would be the ultimate way of removing spam mails.

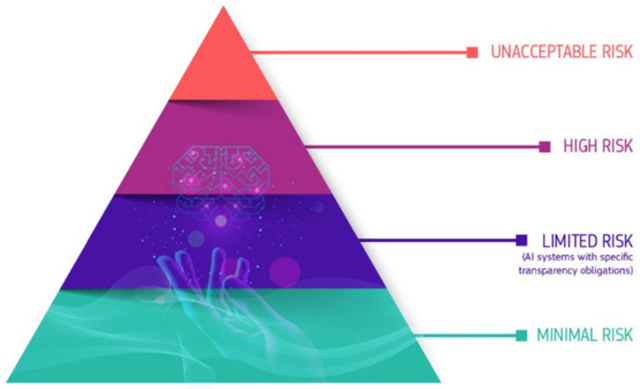


Figure 1: A visualization of the risk-based approach by the European Commission

might be very hard to set all the right rules to prevent things going wrong, and it only has to go wrong once.

> "Our greatest threat is not that AIs turn against us, the risk lies in AIs doing what they are asked to in drastically unforeseen ways."

This example is a variation on the famous paperclip theory thought experiment: a dive into how a super-intelligent AI might cause the apocalypse. To many this example may seem a bit fantastical, but it illustrates how a goal driven AI can cause massive unforeseen issues. A manually implemented solution to the problem "remove as much spam as possible" would likely create a spam filter that filters based on a detection of words, sentences, and email addresses. As humans we understand implicit ethical borders, but an AI simply works within the borders explicitly outlined to it , it has no concept of ethics. Our greatest threat is not that AIs turn against us, the risk lies in AIs doing what they are asked to in drastically unforeseen ways.

So just set stricter rules for the AI, and we will be fine right? Well yes, if we would have set some rules on what the AI was allowed to do and what not we could have stopped it from killing the world. However, the problem is that it

Are we doomed?

Luckily, there are a few things mitigating the danger of rogue AIs. As artificial intelligence becomes more popular than ever so does research into ethical artificial intelligence and explainable AI. Tackling how we apply ethics to AI and how we can understand the process of an AI respectively.

On top of that the levels of intelligence an AI would have to achieve to form such existential dangers to us, are simply not reached. And there is no consensus on when, or even if, such levels of intelligence can be reached. However, there are some that predict that we can reach such levels of AI at the end of the decade.

As long as there is a chance such intelligent AI can be created, we risk existential crises if they are not implemented with an utmost focus on safety. However, we are not doomed, yet.

Figure 2: Security cameras with facial recognition can be used to track everyone's moves.

Regulate before it's too late

While there are agreements between companies to limit the dangers of AI, we will need more than that, because we cannot rely on companies to regulate themselves without supervision. There have been plenty of examples of ethics being disregarded at companies when economic incentives are high. So, with potential risks as big as these it is necessary to set governmental regulations on AI development. The first steps by the European Union are a good start, but they do not tackle the greater, possibly existential, risks of artificial super-intelligence yet.

Yes, it is unsure if we can have an AI smart enough to pose such a danger to us all, but it would certainly be nice to have precautions in place, just in case it is possible.

Acknowledgements

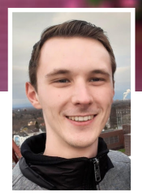
This is an opinion article, there are plenty of arguments to oppose the new EU regulations. The main concern is that it stifles AI innovation, in a continent where we are already behind on its development.

Furthermore, the assumptions that human level intelligence is achievable is a highly debated topic by experts in the field.

Finally, I will recommend a book to anyone interested in similar topics to this: Superintelligence by Nick Bostrom. The book covers many of the topics of this article in much greater detail and outlines some other ways superintelligence might go rogue on humanity.

//Article

LTL Model checking



by: Remco Abraham
Winner ENIAC Thesis Award

Software is playing an ever-increasing role in our day-to-day lives. Visibly, through our use of social media, streaming services, and other applications, but also invisibly, by supporting and controlling critical parts of our society, such as supply chains, water management, medical devices, the power grid, and so on. Whereas a malfunction in our favorite streaming service may result in ruined plans for the evening, the consequences when one of these critical systems fails can be much graver, potentially even risking people's lives.

There are many examples of software failures that have caused serious damages of various kinds, such as financial, societal, or even loss of human lives. A tragic historical example of the latter kind is the set of bugs in the Therac-25 machine. This machine was a radiation therapy machine used in the 1980s and has caused at least six accidents by administering a serious radiation overdose, causing three deaths.

Examples like these illustrate the importance of reliability in software. How sure are we that our software does exactly what it is supposed to do under all circumstances? Depending on the context, the level of the desired certainty may be different. Many software applications can be shown to be sufficiently reliable by extensively testing various scenarios, manually or automatically. However, no matter how extensive, tests are hardly ever exhaustive, meaning that there are almost always untested scenarios. Testing, therefore, reduces the risk of failure but does not eliminate it. If after testing, the risk of failure is still too high when weighing the potential consequences, we need other means to increase our confidence in the product.

Model-checking

One reliable approach entails making a mathematical model of the system in a language for which we can write a formal specification. We can then check the model against the specification using various mathematical techniques such that we can prove that the model adheres to the specification. Assuming the model accurately describes the real system, it provides very strong evidence that the system behaves correctly.

In literature, this technique is known as model-checking. It has seen some great applications already. Usually, the model that is used in model-checking is some sort of automaton which is a graph-based execution model in which the nodes represent the different possible states of the system. We can then use specification languages over these automata to specify the desired behavior of the system.

LTL

Linear temporal logic, or LTL for short, is such a specification language. It is a temporal logic, meaning it is a logic that can be used to write propositions over time. To illustrate, let us first consider the proposition P ^ Q in propositional logic over the atomic propositions P and Q. This proposition has no notion of time. Suppose we have a four-state automaton for all combinations of P and Q, then the proposition P ^ Q can be evaluated for all those states, and only for the state where both P and Q are true will the proposition be true. Now LTL can be used to express statements such as "In the next state, P ^ Q will hold" or "At some point, we will reach the state where P ^ Q holds". We can even express statements such as "Q holds as long as P holds".

To see why this is useful, suppose we are modeling a system for a drawbridge. Let P mean "the bridge is open" and Q mean "the barrier is lowered". Then the statement "Q holds as long as P holds" becomes "The barrier is lowered as long as the bridge is open", clearly a desirable property of a drawbridge. This illustrates that LTL can be used to express the specification of a system.

LTL Synthesis

Model-checking with LTL is extremely useful when we need guaranteed correctness of an existing implementation, but what if we do not have an implementation yet? We would first have to write the specification and build the system to then check whether the system adheres to the specification. This is fine, but what if instead we could automatically create the system from the specification. This would save us a lot of manual work, and having a system that is somehow constructed from its specification should imply that it adheres to it as well. This approach is known as LTL synthesis.

LTL synthesis is not a new technique. The synthesis problem was first introduced by Church in 1957 and was solved in 1969 by Büchi and Landweber. So why is it not widely used everywhere? Among other reasons, LTL synthesis suffers from scalability issues; It is known that the LTL synthesis problem has a double-exponential time complexity. This makes the technique very difficult to apply in practice, however it is not impossible. After all, there are plenty of theoretically exponential problems solved daily. A famous example is the traveling salesman problem which is NP-hard yet has many applications in logistics and the manufacturing of microchips. By clever engineering, we can often make these theoretically difficult problems doable in practice.

> "Whereas a malfunction in our favorite streaming service may result in ruined plans for the evening, the consequences when one of these critical systems fails can be much graver, potentially even risking people's lives."

Binary Decision Diagrams

One technique that can be used to improve the performance of LTL synthesis in practice is by using a smart encoding for propositional formulas. It relies on the observation that any propositional formula can be represented as a so-called binary decision diagram (BDD). BDDs are directed acyclic graphs where each node represents a boolean variable in the formula and each edge represents an assignment of the variable. When used in their canonical form, BDDs can be used as an efficient data model for sets and operations on them, such as union and intersection. Since most algorithms in LTL synthesis operate on automata, and an automaton is essentially just a tuple of sets, BDDs could improve the performance of these algorithms significantly.

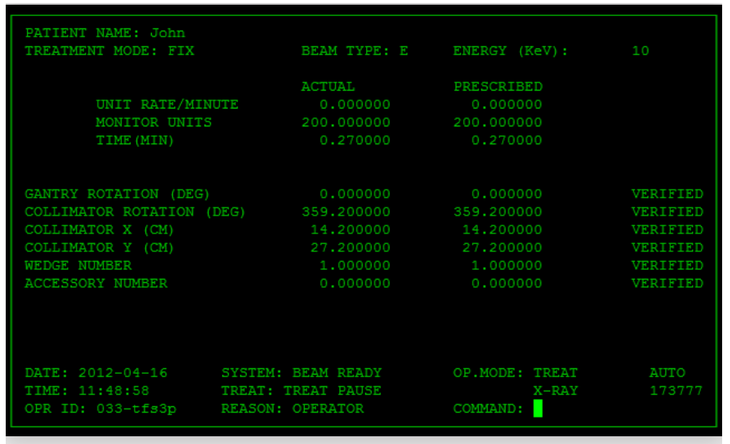


Figure 1: Interface of a THERAC-25 machine.

Research

In my research, I implemented an LTL synthesis tool using BDDs and a recently invented LTL normalization technique. It was then compared against the state-of-the-art. Although my tool was mostly slower than existing solutions, there were interesting exceptions. Most notable was a case where my tool was able to find a solution more than 30 times faster than the winner of the synthesis competition of the last few years.

I submitted the tool to the 2021 synthesis competition as well, and although expectedly it did not receive first place, it also did not end in last place! Considering that I have had a lot less time for implementing optimizations and that the tool was a lot more prototypical than the other submissions, I was quite surprised and happy with the results.

The results of the research have left a plethora of future work. What exactly makes that one example so much easier with my tool than with others? Perhaps we can somehow use this information for further optimization. Also, many other optimizations can still be implemented. It would be interesting to see how far we can stretch this approach. Maybe we can identify structural properties of the automata that can be used to optimize the BDD encoding, or there could be other optimizations that I have fully overlooked.

I invite you to have a look at my thesis for all the details and the mathematics that I could not cover in this article. Feel free to contact me with any questions and be sure to let me know if you are considering the topic for a Bachelor or Master thesis.

About Remco

Remco Abraham finished his master Computer Science cum laude in July 2021. His thesis titled "Symbolic LTL Reactive Synthesis" was graded a 10/10. He was offered a PhD position at the University of Twente but decided to pursue a career outside academia instead. He is currently working as an IT-consultant at NAVARA.

More information, references and the full thesis can be found in the UT database:

https://essay.utwente.nl/87386/

//Puzzle

PRODRIVE TECHNOLOGIES

Puzzle

- The puzzle is to maximize the score of a single player game of Scrabble (Wordfeud)
- Unlimited amount of letters
- You can start everywhere, but every word after the first needs to be connected
- Only allowed to play each unique word once
- Boards are in 25 different sizes of (6x6, .., 30x30), this results in n=6,7,..,30
- A triple word score tile is located on each of the tiles where the row and the column index are a multiple of 5. (e.g. 0,0; 0,5; 0,10; 5,0; 5,5; 5,10 etc)
- The value of a letter is its index in the alphabet modulo 10 (e.g. a=0,b=1,c=2,..,z=5)
- The list of allowed words contains both Dutch and English words and can be downloaded from the website

Scoring

Scoring happens by grading the solution for each n and combining these in a single score.

For each of the 25 values of n, you will get a subscore between 0 and 1. The subscore is calculated by dividing the best score of any contestant for that n by your best score for that n.

Assume you are the first to submit a solution with a score of 50 for n=5. Since this is the only submission for n=5, it is also the best raw score currently submitted for n, resulting in a subscore of 1.0 points.

Another contestant submits a solution for n=5 with a score of 58. Now your subscore is reduced to 0.862 (=50 / 58).

Your total score is the sum of all your subscores. This means your total score is between 0 and 25. The goal is to maximize your total score. *Note that your total score is not fixed! As other people submit solutions your total score might drop, so keep an eye on the submission page.*

Submitting

For each word on the board you want to submit, specify the row index (0-based), column index (0-based), horizontal (H) or vertical (V), and the word.

When you place a word that extends another word, you need to play the extended word. (e.g. the tile before and after your word should be empty or non-existent)

To see the current standing, deadline and to submit your solution, go to: https://puzzle.prodrive-technologies.com