
Francken Vrij
Errors that
every physicist
should like
Francken Vrij
26.2 ERROR
OK

Ready to put theory into practice?

Discover Nedap University, our post-master program for code-loving changemakers!

During one-and-a-half year you will work within one of our business units, developing software together with your team, and follow lectures with other Nedap University participants at the University of Twente, to acquire necessary theoretical knowledge.

Once you've completed Nedap University, you'll be a high-end software developer ready to create Technology For Life.

Check www.nedap.com/university for more information

nedap

Colophon

Editor in chief

Sibren Wobben

Editorial board

Amy Kerdijk
Alexandra Meerovici
Ian Soede
Leon Trustram
Emiel de Wit

Senior Editor

Sjoukje de Jong

Editorial

For this edition we'v come up with a vry exciting theme, ***exact***! ~~It was 'exactly' five years ago that I started working on my first Francken Vrij and jst because of that I thought this would be the perfect opportunity to s tht thme. Th rst f ths pce s jst h n wll vr rd ths nywys..~~.........

Sorry for that, exact was of course the theme of last edition, weird... You might have noticed some things different for this edition. The theme of this edition is 'error', and I think the Francken Vrij keeps glitching out... Are all these mistakes intentional? Or was I just lazy and for once didn't want to do a whole lot of proof reading, who knows? We still hope you'll enjoy!

Address:

T.F.V. 'Professor Francken'
o/c Francken Vrij
Nijenborgh 4
9747 AG Groningen
The Netherlands
Telephone number: 050 363 4978
E-mail: franckenvrij@professorfrancken.nl

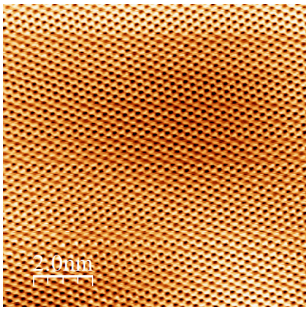
General:

Advertisers
Nedap$_{2}$, Schut$_{28}$
ISSN:
2213-4840 (print)
2213-4859 (online)
Edition and circulation:
June 2022, 100

Special thanks to:

Jelle Bor, Antonija Grubišić Čabo, Luc Cronin, Nea Isla Kauko, Jasper Pluimers, Gertjan Pomstra, Petra Rudolf, Robbert Scholtens, and Bradley Spronk

26.2 Edition



Chair's Preface

By Ian Soede

In my opinion, one of the most surprising facts of nature is that there are some quantities we can never know exactly. Especially on the most fundamental level, according to Heisenbergs uncertainty relation, we can very well know how badly we know things according to the size of our system. Somehow it seems there always is some uncertainty, a margin of unknown, an error. The error does not prevent physicists from doing physics, a lot of information is hidden in where something might be and what momentum it might have.

From what I have learned over the past year, this also applies to Franckenmembers. The uncertainty in the amount of alcohol a group of Franckenmembers has consumed multiplied by the uncertainty in how much of a mess they'll make during the evening, must be greater than or equal to 37 (which follows from obvious reasons):

$$\Delta V_{alc} \Delta N_{mess} \geq 37$$

Now you might think, we can know exactly how much someone has drunk right? That might seem the case, but you never know whether the board has invited them for a 'leermoment' or whether the G13 beer did exactly contain 5,9% alcohol. And if you are quite sure how much alcohol was consumed by a Franckenmember, you have paid too much attention to one person and cannot also be sure what shenanigans the rest have done. It is a fundamental and inescapable error in managing Franckenmembers, and one that is only experimentally found when it is too late.

News of the association

By Sjoukje de Jong

After there was a full lockdown again, it was now time to fully open up everything. Our room is now open for all possible events, which might not be better for my grades but it is better for my mental health! This means that there have been many events, including on-site excursions! I think many of these changes have caused an error in my head at some point, but hopefully, this is the last time that Covid related brain errors occur to me!

Buixcie announcement 2022

The next destination for Buixie was announced, this year we will be going to France! The announcement was held in a GatherTown environment where you could join in groups. Each group got groceries to cook a typical meal from a certain country. When the meal was cooked it was announced with a brilliant video that France will be our next destination. We are scheduled to go to Paris, Marseille, and Iter.

Ice skating with Francken

As it was winter, which makes people want to go ice skating, we decided to do just that! We got a nice clinic from G.S.S.V. TJAS, the sports association for ice skating. We were taught different techniques, which we could then try out on our own on the ice. In the end, it was time to do some

matches against each other, I am not very proud to say that as a Frisian person I lost against a non-Frisian person. Other people of course got to show their skills in this part which was nice to see.

Thales excursion

The first on-site excursion after all the lockdowns finally happened! By train and bus, 15 Franckenmembers traveled all the way to Hengelo where they got to see a company in real life. They first got a general presentation about Thales in general and about the radar systems that they develop. After a nice lunch, they got a tour of the design and assembly of the radar systems. After that, they also got to see some things that are developed for the army. Last but not least they got an in-depth presentation on how radar systems work. When they got they were treated to a nice pizza.

Sympcie announcement 2022

As it was time to announce the theme of the upcoming symposium, a nice borrel was organized. This was at Cafe Eureka with our lovely bartender Ricardo! The theme was announced by a very long video, the theme of the next symposium will be 'Go with the Flow'! I am sure there will be some nice talks on fluid mechanics, where this knowledge may be applied after the symposium.

H-GMA

This GMA was also on campus after having held GMA's online or at different locations for a while. While the board presented all the important stuff, the members tried to distract them by writing tons of motions. Discussions about what can and cannot touch the flag, whether the Polycon could see you, or what food you should order were a few amongst more important discussions. But in the end, it was a successful GMA.

Francken's got talent

Fraccie was curious about how talented our Mooie gekken are, especially in the field of music. While completing different challenges the most talented participants could be found. These challenges included guessing reverse played songs, figuring out text that had been through google translate a lot, and a category where only the start of songs could be heard and had to be guessed. In the end the most talented Franckenmembers turned out to be our Chair Ian together with Eelco who got a great price!

'Back to the 80s' with Sjaarcie

As you know, one of the two jobs of Sjaarcie is to host an epic party! I think they were pretty successful in doing so, as almost the whole basement of Club Kiwi was filled by (mostly) sjaars. Everyone was dressed up in their prettiest 80s outfit and could show off their moves on the floor or even on the pole. Our kalashnicup skills were also tested during some fun games, let's just say I was not very lucky.

Francken Friday Lecture from Ralf Mackembach

Yes, you heard it right, the boy that won the Junior Eurovision song contest grew up to do a PhD in nuclear fusion! As we are very curious about what this entails, we invited him to give a lecture about his research. While enjoying snacks and drinks we found out that next to his very interesting research, Ralf also thinks that there should be more emojis used in lecture slides. After his compelling lecture, there was of course a borrel in our members room. We were about as disappointed in his height as in his drinking abilities, but these things might be correlated.

Belsimpel Inhouse day

Our second real-life excursion was a bit closer to home, right in the center of Groningen. We visited the main building of Belsimpel where they showed us why their phone company could also be considered a tech company. After they presented the ins and outs of their company we got to work on a data science case, where we could see what a job a Belsimpel might look like. When we were done we got free pizza, drinks, and a great goodie bag followed by a nice tour through the whole building.

Errors that every physicist should like

By Jelle Bor

Theoretical physics and applied physics are often quite different diciplines, however we both encounter errors often. Did you know there is a function for that?

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

In statistics, the error function has the following interpretation for non-negative values of x: for a random variable y that is normally distributed with mean 0 and standard deviation $1/\sqrt{2}$, $erf(x)$ is the probability that y falls in the range $[-x, x]$. The error function is antisymmetric, $erf(-x) = -erf(x)$, and $erf(0) = 0$ and $erf(\infty) = 1$.

Some people like errors, others don't; this reminds me of the AIVD christmas puzzle. Can you solve this integral for me without using a computer?

$$\int_0^{e^{-e}} e^{e^{e^{e^{e^{e^{e^{e^{e^{e^x}}}}}}}}} \dots e^{e^{e^{e^{e^x}}}} e^{e^{e^{e^x}}} e^{e^{e^x}} e^{e^x} e^x dx$$

Please email you answer to jel... and you can ...

Damn it, looks like my text got stuck - "Quit kernel"

The name error function and its abbreviation erf were proposed in 1871 by Glaisher on account of its connection with the theory of Probability, and notably the theory of Errors. Gaussian functions are often used to represent the probability density function of a normally distributed random variable with expected value μ and variance σ. Then the Gaussian becomes of the well-

known form:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

Glaisher calculated the chance of an error lying between p and q as:

The error and the complementary error function, $erfc(x) = 1 - erf(x)$, occur for example in solutions of the heat equation when boundary conditions are given by the Heaviside step function.

... Error: could not found Heaviside step function, define new variables accordingly...

Besides errors in general, another problem is propagation of errors, i.e. something has an error and you just plug it in another function. Luckily, I am a theorist and do not come across this often, but you should all know how to handle such situations. Do you remember the variance formula, neglecting correlations or assuming independent variables,

$$\sigma_f = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2\sigma_x^2 + \left(\frac{\partial f}{\partial y}\right)^2\sigma_y^2 + \left(\frac{\partial f}{\partial z}\right)^2\sigma_z^2 + \ldots},$$

where σ_f represents the standard deviation of the function f, σ_x represents the standard deviation of x, σ_y represents the standard deviation of y, and so forth. This formula is commonly used among engineers and experimental scientists to calculate error propagation, but one can also just open your Physics Lab I reader to look it up for specific formulas. Have fun.

Now, let's just quickly look at the worst case of errors: errors which unfortunately give the right answer, so you don't know that you did something wrong (hopefully you see). Here's an example:

$$\begin{aligned}\int \frac{1}{x+1}dx &= \int\left(\frac{1}{x}+\frac{1}{1}\right)dx \\ &= \int\frac{1}{x}dx + \int 1dx \\ &= \log(x) + \log(1) + C \\ &= \log(x+1) + C\end{aligned}$$

As these are fun, a few more:

$$\log(1+2+3) = \log(1) + \log(2) + \log(3)$$

$$\frac{d}{dx}\frac{1}{x} = \frac{d}{d}\frac{1}{x^2} = -\frac{1}{x^2}$$

$$2^5 9^2 = 2592$$

$$\frac{95}{19} = \frac{5}{1} = 5$$

Personally, I often encounter errors when programming, which is yet another form of errors and probably the most annoying one. Whether you use Python, Mathematica, Matlab, Fortran, etc. or even object oriented like C: it all comes down to continuously solving errors. Hè isn't there a function for that? - No, just Google it.

It seems that I still need some debugging to do for this article...

Life after Francken

By Jasper Pluimers

While engineers are content with a reasonable error that fits the requirements of the assignment, we theorists like to do everything exact. This does not me... Wait a second, wrong column. It has been a while since I wrote something for the Francken Vrij, you can remember me from my time in the editorial board or, later, as the theorist. As this is probably the last time the Francken Vrij committee had to remind me of the deadline 8 times over a month, please enjoy my Life after Francken. From the day I started studying until about a year before I left Groningen I thought I would stay a theorist forever. I often said things like: "I prefer to think in N dimensions" and "Real stuff is boring". My opinion on this started to change a bit when I was writing my master thesis and I discovered I did not really enjoy the academic research itself, I just really enjoyed learning new things. So what do you do if you ignored all career events during your studies like a good physicist? You go to lunch lectures for the content instead of the lunch. It did not take a long time before Jasper Compaijen, also a former theorist, came to Groningen to tell us about the company he was working for: Nedap. At Nedap retail he was working on electronic anti theft systems that works with RFID and they also have a software development traineeship that targets beta graduates. As I used programming as an excuse to not write my thesis all the time I thought: "Why not?".

Software Development

During this traineeship you follow Computer Science lectures at the UT in Enschede and you start working at the same time. I like the idea of it because contrary to some other traineeships you actually follow lectures at a university and you get a long time (1.5 years) to actually learn stuff. I don't want to make this a recruitment piece, so if you have any questions about this please send me a message.

While you touch some programming during your studies I discovered there are roughly three levels of programming. First of there is the first contact you make with programming during your studies. It might be in Matlab, Mathematica or if you are lucky in python. The focus is on either passing the programming assignment or maybe writing something that helps you with another course. Most scripts you write will be so messy you won't even know what the variables were when the student assistant asks about them in the next tutorial. There is at least a nested loop of 8 levels and every time you run the program you cross your fingers it does not crash. The advantage of this stage of programming is that it is quick and the responsibility is low. It only has to run until it gets graded and if you forgot about it in a month nobody will bother you. The disadvantage is the somewhat slow learning process, the error messages won't make a lot of sense yet and sometimes the computer seems to do stuff in an illogical way (it really is you who is illogical). You will notice this the next year when one of your fellow students asks if you still have your programs and you answer with: "yes, but...". Someone named Mark will tell you that you have to commit to learn Git to do all version management and that if you use floating windows on your PC you are a loser and you ignore it.

If you did not totally hate this experience there is a good chance you will revisit programming with a bit more care. Maybe there is some privacy invading tracking device you want to install in the Franckenroom or maybe you have to write something for your thesis. You won't be in Groningen forever so you bear some responsibility for others right now, every time you write some code, you have to think: "If someone sends me a message saying my thing broke, can I understand what I tried to do and help fix it". There are some more complex concepts you get familiar with like classes, objects and tests. Someone named Mark will still tell you to use Git for your version management and you might check it out, but you do not really understand what you are doing and why it would be useful. At this stage you start learning faster and faster, which is fun. You will learn to read documentation, check out some programming blogs and decide that either object oriented or functional programming is the best thing. Ultimately you start understand more, but question even more than that.

For some people the journey will end here and that is fine. A lot of DIY projects can lie ahead of them and it will prove to be a useful skill your whole life. The next step is when you want to (or have to) write code with others and used by others, this is when programming turns into software development. A lot of effort goes into making sure the software you write can still be read and understood by colleagues in 10 years. It is not enough anymore to know yourself that the program is working, you will have to prove it by writing tests for your code. Designing and writing documentation for the programs that you build will take at least as much time as the programming itself which can be a bit tiring. When you look back at this code in a few years you will appreciate the thought you put into the choices that you made and you finally understand why Mark pushed that Git thing so much. This is a stage that you will not grow out of and will never stop learning new things about, which is why I like it so much.

Nedap

At the Nedap Retail we mostly work with RFID tags in clothing. RFID tags are small chips hold a unique number and can send that unique number to you when probed, similar to the NFC system on your phone. We use radio waves to both power and transfer data to and from these RFID tags and use that for two main propositions: A stock management system and an anti theft system. Everyone who reads this has walked through this anti theft system multiple times in their lives, for example at the entrance of a Decathlon or H&M store. We designed all hardware in-house and the team that I am part of is responsible for all firmware that runs on that hardware. This firmware varies from some high level code in the installation wizard that we made to

very low level C++ code that handles the high-throughput observations from the RFID-reader. Recently we decided to re-write most of the code in Rust, a fairly new programming language which promises C level speeds but has some pretty strict rules in what it allows to compile. This has a steep learning curve but helps you to write safer code and completely prevents some type of bugs from appearing! If you would like to know more about Rust, check out the website (https://www.rust-lang.org/) or send me a message.

Luckily physics is not completely absent from my life, as there are quite some challenges left in optimizing these anti theft gates. It is easy to detect (almost) all tags, but it is quite hard to only observe the tags that move outside and not the once that are just close to the entrance. Just like in Astronomy you have to separate the things you want to observe from the noise, but unlike Astronomy the length of our wavelengths are not that different from the distance to the targets. Retail environments are full of excellent radio reflectors and absorbers, some inanimate like metal plating on walls, but also animate ones like customers. This combination of Physics and Software Development will keep me busy for a long time to come.

One of the disadvantages of finding a job is that you will have to move within reasonable distance of the company as well. I always thought Groningen was a pretty remote place, but Groenlo is on a whole other level. Luckily due to corona it is getting more and more common to work from home, such that I do not have to make the journey to Groenlo every day. On top of that I don't have to miss Francken that much as Nedap Retail currently employs four mooie gekken, while not as many as a certain chipmaker company it has a lot of potential.

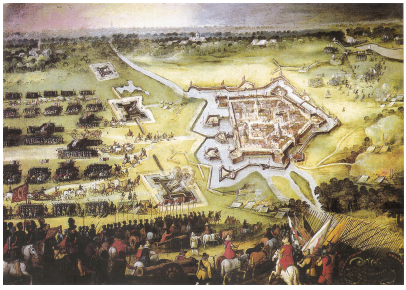


Figure 1: Groenlo, famous because of the slag om grolle and as the birthplace of Grolsch

The biggest lesson I learned over the past few years is that even if you think you know what you want and what you do not want, trying out something new can make you realize there is a lot more to do and learn. It turns out that real stuff does not have to be boring and it can motivate a lot to see the things you make being used by people all over the world.

Comic

By Bradley Spronk

error 404:
comic not found

Corruption

By Robbert Scholtens

Data is good; data is right; data ... is life. But corrupted data certainly is none of these things. Unfortunately, we live in a world where data is corrupted all the time, be it due to transmission loss, spontaneous electron flips, or even cosmic rays. If unable to handle the inevitable corruption of data, pretty much all of our digital advancements would be moot - highly inconvenient, if you ask me. So, how can we handle corruptions like these?

To give you a flavor of coding theory, which is the mathematical discipline concerned with this handling, I've divided this article into two parts. In the rst part, I will treat an example adapted from Cecilia Salgado's recent talk for a very specic setting, and in the second I introduce the proper terminology and present a generally applicable result.

A spirited example

The setting is that you wish to order one of eight brands of beer from your local pub. But since the barman is particularly stubborn, you have to submit your order as an n-bit message, from which he should be able to determine which beer you want. (Also not unimportantly, you can tell him beforehand how he should make his determination.)

Initially you might just submit the 3-bit label belonging to your beer brand of choice -message 1 in Table 1- but this would not be wise. Namely, if along the way one of the bits is flipped, the barman will receive *at least* 3 bits (for there are 8 = 23 choices), the barman will give you the wrong beer.

A different tactic would be to duplicate your message, sending your order "twice," effectively. However, again you have the problem you confuse the barman if one bit is flipped along the way: if he receives for instance 010 011, did you order Dors or Jupiler? He's got no way to tell, and so

Brand	Label	Msg. v1	Msg. v2	Msg. v3
Hertog-Jan	000	000	000 000	000 000
Heineken	001	001	001 001	001 011
Dors	010	010	010 010	010 110
Jupiler	011	011	011 011	011 101
Amstel	100	100	100 100	100 101
Grolsch	101	101	101 101	101 110
De Klok	110	110	110 110	110 011
Bavaria	111	111	111 111	111 000

Figure 1: Table of beer brands, with associated labels and messages.

you have 50% chance of getting the beer you actually want. We're not really getting places ...

Two observations at this juncture: i) we increased the amount of information sent (six instead of three "strictly necessary" bits), but also ii) we increased the robustness of our transmission (50% of getting the correct beer with one bitflip, instead of 0%). Keep these observations in the back of your mind; they'll be made concrete later.

Returning to the problem of beer-ordering, we should be cleverer about how we compose our message. An example of a cleverer messaging scheme is the last column in 1. How were these messages formed? We use the XOR operator, and follow the guide in Table 2. The bits added in this way are known as *parity bits*. The beautiful thing about this construction is that whenever a message is now received, and one bitflip has occurred, we still receive the beer we ordered with 100% certainty!

Short of giving you a formal proof of this fact, let a proof by example suffice. Suppose you wish to order Grolsch, so you send the message 101 110, which is received by the barman as 111 100. Since this received message does not correlate directly with a beer, the barman concludes a bitflip must have occurred. Upon $\oplus$ comparison to the valid messages -the last column in Table 1- the barman sees there is only *one* which differs from the order in a single place: Grolsch. As such, the barman concludes that that was the beer you *actually* intended to order, and gives you your refreshing beverage. Amazing!

Okay, this was obviously a very playful example, but it does exemplify a potential avenue for research. This is what is done in the mathematical discipline of *coding theory*, and it is big business due to the need for data robustness with which I introduced this article.

The theoretical bit

It would be a shame not to give you a few more technical details, to see what's going on under the hood and furnish your (inevitably piqued) interest. The "sea" of potential binary messages that could be received is the *state space*, and in our case it had dimension $n=6$. In contrast, the binary messages we could send are known collectively as a *code*, and for us that had dimension $k=3$ (for we needed 3 bits only in order to characterize the entire message, namely the label of the beer, and then the other three were found from Table 2).

The final concept is that of a "distance" between messages. The one that is used most often is the *Hamming distance* d_{ham}, which simply counts in how many places the messages are different:

Over the code C we can then nd the minimum distance d between all the messages:

In our case, it can be veried that the minimum distance $d = 3$. This means that any element of our code diers from any other element in at least three places.

Evidently, we would want to get the *most* amount of data robustness for the *least* amount of additional bits needed. That is to say, we would like d to be as *large* and n to be as *small* as possible: d regulates roughly how many bitflips we can handle, and n (or rather, n-k) how much additional information/communication

$$d_{\text{ham}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_j x_j \oplus y_j.$$

is added on top of the "necessary" k. As you can imagine, these three quantities

$$d := \min_{\boldsymbol{c}_1, \boldsymbol{c}_2 \in C; \boldsymbol{c}_1 \neq \boldsymbol{c}_2} d_{\text{ham}}(\boldsymbol{c}_1, \boldsymbol{c}_2)$$

are interrelated in such a way as to regulate their sizes. The inequality is known as the *Singleton bound*[4], given by

So you see, we can't get arbitrarily large robustness with arbitrarily small sizes: to improve one, there is a cost in the other. This we saw in action in during the example: with the addition of three bits, we were able to make a more robust message than without them. Furthermore, in our example above $n=6$ and $k=d=3$, so the inequality $3 \leq 6-3+1=4$ is fulfilled-phew! However, this also suggests that we could've potentially designed a better code: either $d=4$ or $n=5$ would also have been allowed. That such a more optimal code exists is *not guaranteed*, but I leave it as an exercise to the reader to either design a code that has $n=5$ or $d=4$,

$$d \leq n - k + 1.$$

or show that such a code does not exist!

Noteworthy also is that in the limit of large messages, $n/d \to 1^+$, so theoretically most of our sent message could be corrupted and we could *still* recover the information we wanted to send. This would work greatly in channels that are highly unreliable and only let a little bit of the actual message through.

Some concluding words

As you can imagine, I've only been able to scratch the surface of error detection/correction, which is even but one subfield of coding theory. Nevertheless, I hope that this has given you a small taste of the power that mathematics has in terms of making our information-driven world robust and reliable. I've also included some material for further reading, but I would especially highly recommend the video by 3Blue1Brown[2] as a good introduction to the field!

References

1. Wikipedia, Reed-Solomon error correction, https://en.wikipedia.org/wiki/ Reed%E2%80%93Solomon_error_correction
2. Grant Sanderson, How to send a self-correcting message (Hamming codes) (2020), https://www.youtube.com/watch?v=X8jsijhllIA&abchannel=3Blue1Brown
3. Guruswami et al, Essential Coding Theory (2022), https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/web-coding-book.pdf
4. Henk van Tilborg, Coding Theory: a rst course, https://www.win.tue.nl/~henkvt/images/CODING.pdf

Defects

By Antonija Grubišić Čabo and Petra Rudolf

In our university lectures we most often deal with perfect, infinite crystals. In reality, however, no material is (obviously) infinite, or perfect. Real materials always host some kind of defects, and the nature and amount of defects will differently affect diverse properties of the material. Some defects occur naturally in the crystal, but sometimes we also introduce defects on purpose to obtain a specific property in a material. For example, transistors in our computer chips are built of p-doped and n-doped silicon but this doping is man-made through introduction of boron or phosphorous impurities in pure, clean silicon that is an intrinsic semiconductor ("neutral" in doping). We can define defects in materials based on their geometry and shape into 0D, 1D, 2D and 3D defects. 0D defects are or point defects can be vacancies (missing atoms) or impurities (as in the case of p- and n-doped silicon). Line defects are 1D defects that come in the form of line and screw dislocations. In real materials, we typically have a mix of the two. External surface or 2D defects as the name says appear on the surface of the crystal. This most often happens because atoms on the surface have very different environment from the atoms in the bulk of the crystal, and in order to minimize energy they create surface reconstructions (change crystalline stacking on the surface) or form unsaturated bonds. And lastly, 3D defects propagate through the whole volume of the crystal and can be stacking faults, voids, crystal twins or grain boundaries, to name a few. This was all for our "regular" 3D materials, but what happens when we look at 2D materials and role of defects?

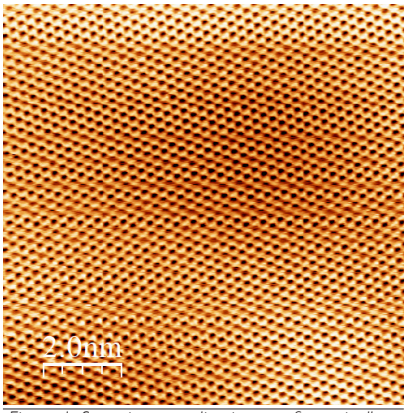


Figure 1: Scanning tunneling image of atomically resolved bilayer graphene on SiC showing a perfect lattice of graphene with no point or line defects.

Defects in 2D Materials

Well, it turns out that defects are even more important in materials that can be considered a pure surface! Not only that, but presence of defects can also have very different effect compared to their presence in 3D materials. As an example, we will present a couple of examples of defects in most famous 2D material – graphene. Graphene, shown in Fig.1, is a single sheet of graphite, that can be either grown on a substrate, or exfoliated with a bit of a help from a regular sticky tape-and is really a wonder material. Electrons in graphene have no mass, so they can move really fast, and graphene itself is a very strong and elastic material, much stronger than steel. Now, what can defects do in graphene? Let's start by looking into two examples of surface modification made by hydrogen, as graphene is a pure surface.

I) A very special and cool effect can be achieved by binding hydrogen atoms to every second carbon atoms in graphene-it can make graphene magnetic[1]. Now why is this special? Well, neither carbon that makes graphene, nor hydrogen are magnetic materials, but if we combine them in this particular way they form a magnetic system. This is not something that you can observe in 3D materials!

II) If we grow a periodic lattice of hydrogen on graphene we can open a band gap in the electronic structure of graphene-and band gap is important if we want to have graphene transistors. Not only that, but depending whether the hydrogen lattice is perfect or has defects, i.e. missing hydrogen structures, the size of the band gap can be different[2].

Both of these examples show how graphene can be improved by defects, so it might seem that all defects in graphene are useful - unfortunately this is not the case. Just as in 3D, defects in 2D materials often degrade their properties. One of common defects seen in graphene, especially one that is grown and not exfoliated, is the presence of grain boundaries. These are the borders where grains of graphene of different orientation merge, and in this border region amount of disorder is relatively high. As a consequence, this reduces mobility of charge carriers in graphene, disadvantageous if we want to make graphene electronic devices, and makes it more likely that diffe-

rent kinds of atoms and molecules can pass through graphene-problematic if we want to use graphene as a sort of a protective coating[3]. For this reason, a lot of effort is invested in developing methods of growing graphene (and other materials) with as few defects as possible, but also studies of what defects actually do-because they are not all the same-are extremely important.

Defects in Nanostructures

As an example for this kind of defects we cite part of the work Sadaf Akbar, now lecturer for the Physics labs, did for her PhD project. She studied stannic oxide (SnO2), a wide band gap semiconductor that exhibits both relatively high electrical conductivity and insulator-like transparency in the visible range and demonstrated how incorporation of a non-magnetic dopant, zinc, in SnO2 nanoparticles enhances the ferromagnetic response very significantly but in a limited range of zinc concentrations[4]. This has to do with the complex interplay between nanostructure, defect formation and consequent ferromagnetism. Sadaf found that the morphology of the nanostructures varies with zinc concentration and that the strongest ferromagnetic response comes from nanostructures with nanoneedles on their surfaces. This is related to the role of the surface planes of the nanoneedles in stabilization of ferromagnetic defects.

Defects are often created when self-assembled monolayers of organic molecules with a functional group are used as active layer in devices. This happens for example for ordered arrays of molecules that switch conformation when irradiated with light. We showed recently[5] that self-assembled monolayers with photochromic moiety degraded rapidly when switched with light in

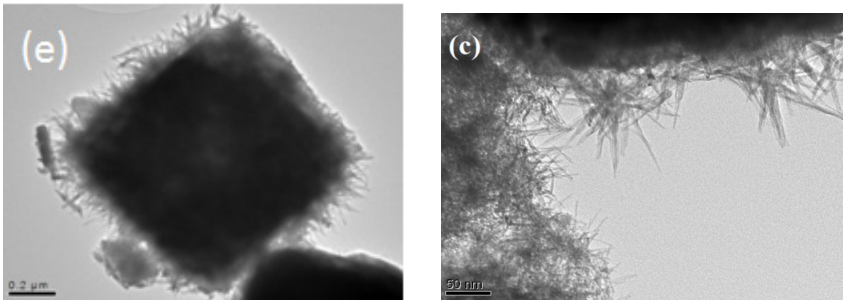


Figure 2: Transmission electron microscopy images of SnO2 nanoparticles doped with 4 at.% Zn, which present nanoneedles on the surface.[4]

an environment with even only small relatively humidity while under dry conditions no chemical degradation is observed and the switching process is reversible over at least 100 cycles. In fact, a photoexcited molecule can be reactive towards its environment even when the same molecule in its ground state is inert. This can cause fatigue and irreversibility of a device. Our results highlight that by creating the right conditions in which photochromic compounds are utilized the device-relevant functionality of surface-bound switches can be preserved.

Concluding remarks

No material is perfect as in our textbooks, as all real materials host different kinds of defects in their lattice. What defects do in the material and how they affect it depends on a lot of things, mainly on type of a defect, quantity, but also on the kind of material as well. Defects in 3D materials are (occasionally) very different form defects in 2D materials and defects on nano-scale as quantum effects are more important on small length scales. Not all defects are detrimental, and sometimes we want to artificially introduce defects to change material in a way that makes it more useful for us. We've used this extensively in silicon industry, but also, for example, in jewels industry-coloured diamonds are just "regular" diamonds with impurities. Most importantly, if you want to tailor material for a specific purpose, you need to know a lot about not only the material in question, but also about defects and what they can do. Defects can be created for example in a molecular layer through interaction with light because photoexcited molecules can react with water or oxygen in the environment and this causes detrimental effects in devices. This makes defects one of more important topics in studying and making designer, functional materials.

References

1. H. González-Herrero et al., Science 352, 6284 (2016)
2. J. Jørgensen, Antonija Grubišić-Čabo et al., ACS Nano 10, 12 (2016)
3. L. Kyhl, S. F. Nielsen, Antonija Grubišić-Čabo et al., Faraday Discussion (2015)
4. S. Akbar, S. K. Hasanain, O.Ivashenko, M. V. Dutka, N. Akhtar, J.Th.M. De Hosson and P. Rudolf ", RSC Advances 9, 4082 (2019)
5. S. Kumar, S. Soni, W. Danowski, I. Leach, S. Faraji, B. L. Feringa, P. Rudolf, R. C. Chiechi , The Journal of Physical Chemistry C, 123(42), 25908-25914 (2019)

Puzzle

By Gertjan Pomstra

Fixing the coffee machine. Every morning you need your cup of coffee to get your day started and thus you power up your favourite coffee machine. But it has got issues and exactly today it refuses to provide you with the fuel you desperately need. So, you decided to turn the thing inside out. As a real TN'er you unscrew the panel, remove all the components of its circuit board, check conductivity with your multimeter and fix a broken wire. That should do the job.

Before you can celebrate with a scalding hot cup of sludge, you must reassemble the coffee machine again. In your enthusiasm of fixing the wiring, you have forgotten in which slot every component should be placed. To make matters even worse, you don't know which two keys you need to reset the coffee machine. Great! Luckily, you have found two pieces of paper underneath the coffee machine which guide you in this unnecessary complex process. Weld the components in the correct slots and reset the coffee machine with the right keys so you can finally get your day started. Good luck!

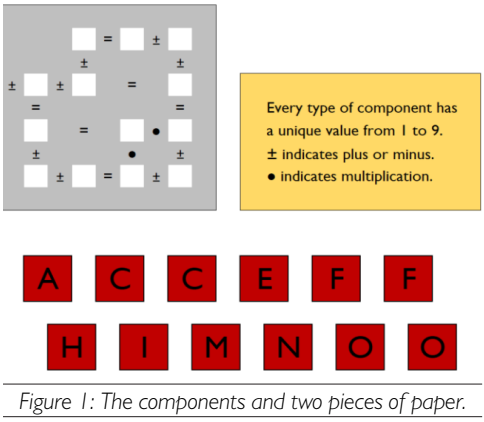


Figure 1: The components and two pieces of paper.

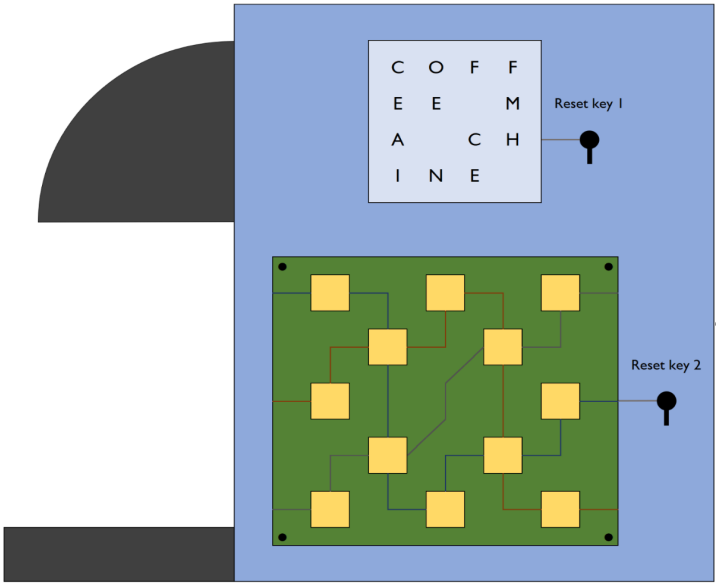
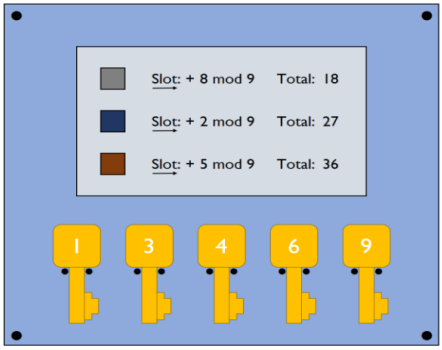
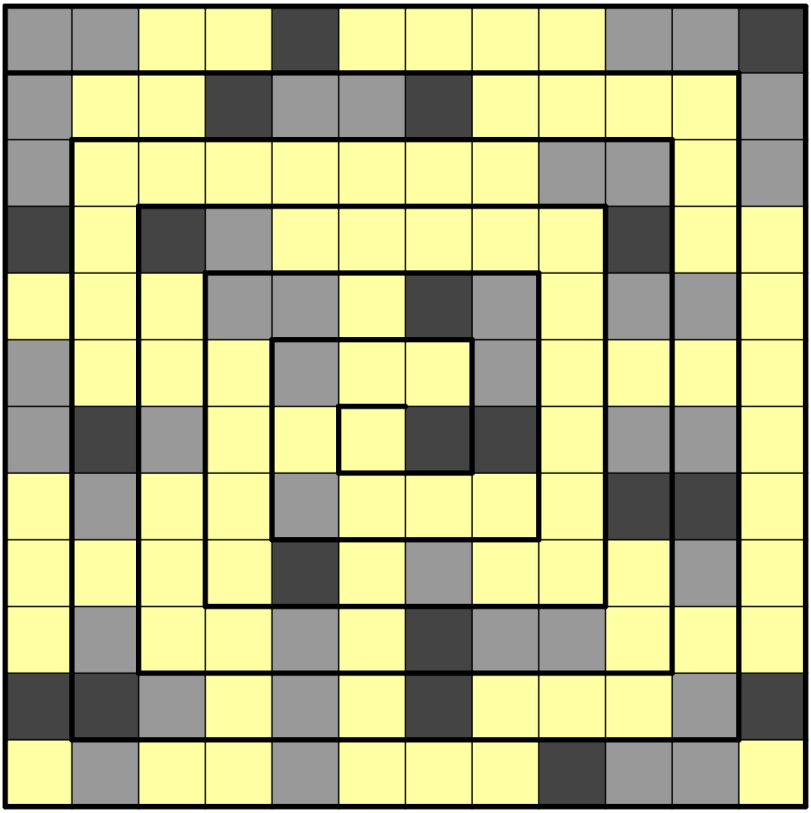


Figure 2: The dreadful coffee machine.



The solutions to last edition's puzzle:



It's all in your head!

By Nea Isla Kauko & Luc Cronin

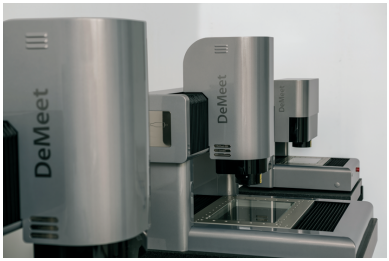
Picture this: it is 1:30am and we're crying tears of joy over FaceTime. What had happened? We finally passed all test cases on Themis, our screens turn green and our eyelids are heavy from the hours spent debugging that one Algorithms and Data Structures lab assignment.

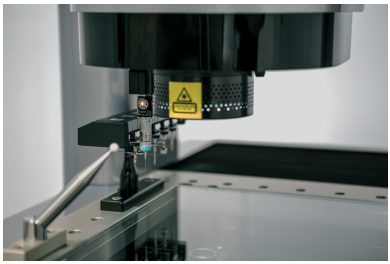
When we logged out of the gather.town tutorial space, our task seemed doable and our code was giving correct output most of the time. After some final fixes and lay-out changes, we compiled the program on our machine and used all available test cases to check our output and it was correct. So, the intuitive next step was to upload the solution to Themis and be done for the day, but Themis rejected our submission!

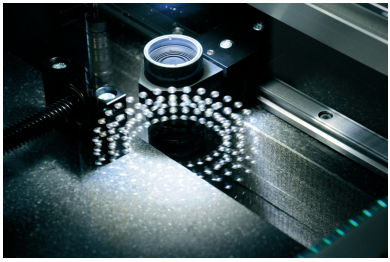
A great amount of professional debugging, including sarcastic print-statements and (un-)commenting later, we still couldn't find our error and at 11:30pm we emailed our TA. Around half an hour later, we get a reply (shoutout Ben!) and he fixed the makefile and told us to try again. Sadly, this didn't help our problem, which we were still pondering.

As a last resort, we went through the header file of our project, and there it was! We had absentmindedly commented out one of the functions in the list. Somehow, it didn't affect the compilation on our laptops, but was a problem for Themis. Hours of dissecting our code all due to two forward slashes…

We slapped our working solution with the updated header file into Themis, and with great relief watched the bars turn green and satisfied. A joyful teary glance at the clock revealed 1:30am - bedtime.







Schut Geometrical Metrology (Schut Geometrische Meettechniek bv) is an international organization, founded in 1949, with five offices throughout Europe, specialized in the development, production, sales and service of precision measuring instruments and systems.

Products developed and produced by Schut Geometrical Metrology are the 3D CNC coordinate measuring machines DeMeet in video as well as multi-sensor model. The DeMeet 3D CNC measuring machines provide automatic, user-independent quality control with measuring results traceable to the international length standard.

Because we are expanding, we are continuously looking for enthusiastic team players to strengthen our company. If you want to work in a company that values people with ideas and initiative, with a transparent company structure and informal, no-nonsense company culture, then Schut Geometrical Metrology is interested to get in touch with you. Employees working in our technical sales, software support and development departments have an academic background.

For various departments we are looking for enthusiastic colleagues with a flexible attitude. The job is an interesting mix of working with people and advanced technology.

We are interested to get in touch with:

- Software Developers (C++)
- Technical and Software Support Engineers
- Mechatronics Engineers
- Technical Sales Engineers
- Service Engineers

You are welcome for an exploratory conversation, an interview or consultation about the possibilities of an internship or graduation project.

You can contact us by e-mail Jobs@Schut.com (“job” as subject) or send your resume and letter to Schut Geometrical Metrology, Duinkerkenstraat 21, 9723 BN Groningen, The Netherlands.

Please have a look at our vacancies on our website:

Jobs.Schut.com

APPROVE for DeMeet

Schut

189_778-2018.02