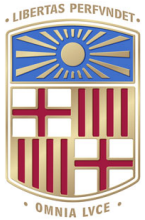
Model checking and formally verified software for temporal quantitative regulations

Joost J. Joosten

Universitat de Barcelona

Universiteit van Groningen
Groningen, May 18

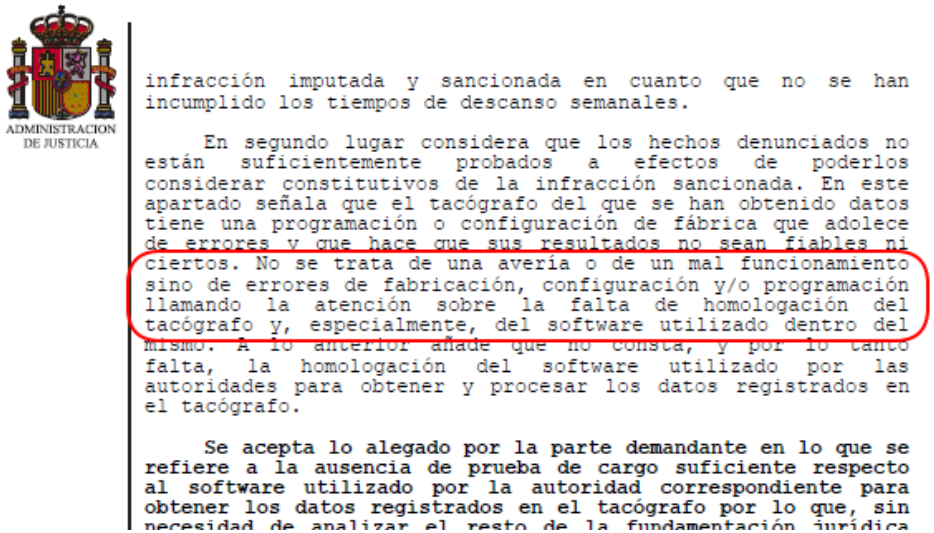
The business model of our research lab



UNIVERSITAT DE BARCELONA

Business focussed: from concrete to abstract

Nice to have or need to have?



ADMINISTRACION DE JUSTICIA

infracción imputada y sancionada en cuanto que no se han incumplido los tiempos de descanso semanales.

En segundo lugar considera que los hechos denunciados no están suficientemente probados a efectos de poderlos considerar constitutivos de la infracción sancionada. En este apartado señala que el tacógrafo del que se han obtenido datos tiene una programación o configuración de fábrica que adolece de errores y que hace que sus resultados no sean fiables ni ciertos. No se trata de una avería o de un mal funcionamiento sino de errores de fabricación, configuración y/o programación llamando la atención sobre la falta de homologación del tacógrafo y, especialmente, del software utilizado dentro del mismo. A lo anterior añade que no consta, y por lo tanto falta, la homologación del software utilizado por las autoridades para obtener y procesar los datos registrados en el tacógrafo.

Se acepta lo alegado por la parte demandante en lo que se refiere a la ausencia de prueba de cargo suficiente respecto al software utilizado por la autoridad correspondiente para obtener los datos registrados en el tacógrafo por lo que, sin necesidad de analizar el resto de la fundamentación jurídica

Sentence Number: 30/2019, CONTENCIOSO/ADMTVO court N. 4 of Valladolid

Law and Code

- Law essentially discretional powers when applied

Law and Code

- Law essentially discretional powers when applied
- Hence, ambiguity is needed

Law and Code

- Law essentially discretional powers when applied
- Hence, ambiguity is needed
- Any automated process and in particular, any automated process in the legal sector need unambiguity

Law and Code

- Law essentially discretional powers when applied
- Hence, ambiguity is needed
- Any automated process and in particular, any automated process in the legal sector need unambiguity
- The programmer needs to disambiguate?

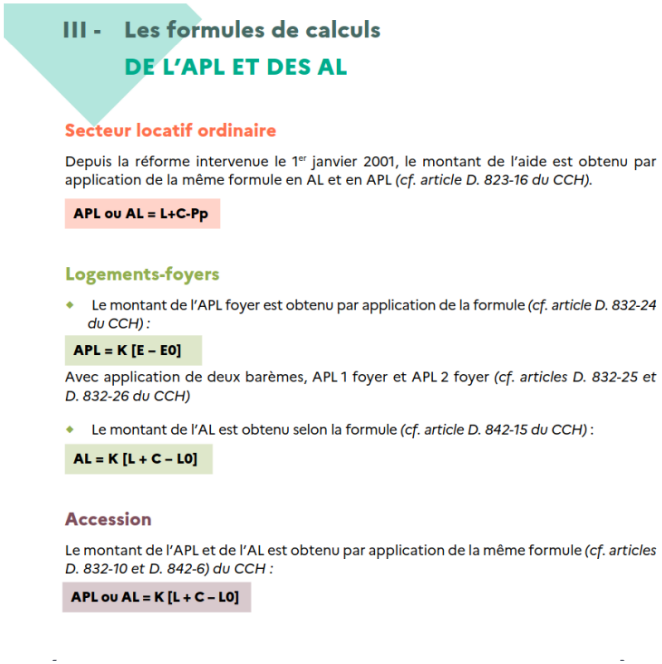
Law and Code

- Law essentially discretional powers when applied
- Hence, ambiguity is needed
- Any automated process and in particular, any automated process in the legal sector need unambiguity
- The programmer needs to disambiguate?
- Can code be law?

Law and Code

- Law essentially discretional powers when applied
- Hence, ambiguity is needed
- Any automated process and in particular, any automated process in the legal sector need unambiguity
- The programmer needs to disambiguate?
- Can code be law?
- And, what if there's an error in the code?

Doubtful results



III - Les formules de calculs
DE L'APL ET DES AL

Secteur locatif ordinaire

Depuis la réforme intervenue le 1er janvier 2001, le montant de l'aide est obtenu par application de la même formule en AL et en APL *(cf. article D. 823-16 du CCH).*

APL ou AL = L+C-Pp

Logements-foyers

- Le montant de l'APL foyer est obtenu par application de la formule *(cf. article D. 832-24 du CCH) :*

APL = K [E – E0]

Avec application de deux barèmes, APL 1 foyer et APL 2 foyer *(cf. articles D. 832-25 et D. 832-26 du CCH)*

- Le montant de l'AL est obtenu selon la formule *(cf. article D. 842-15 du CCH)* :

AL = K [L + C – L0]

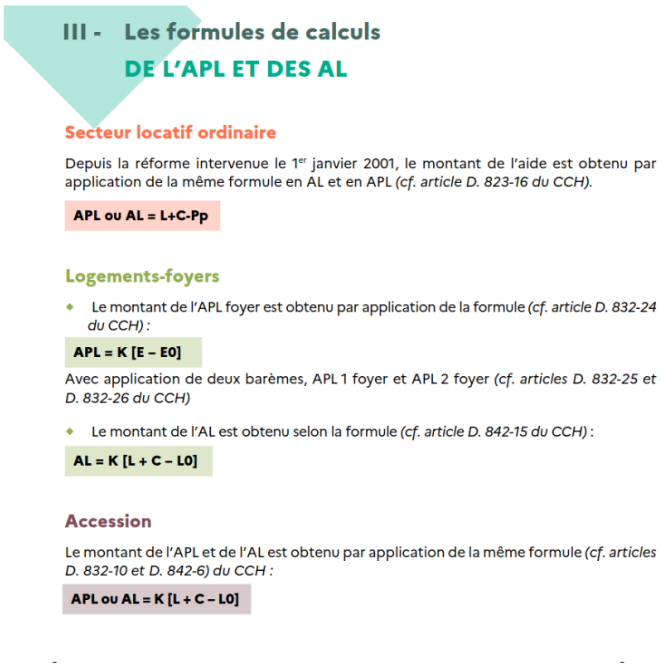
Accession

Le montant de l'APL et de l'AL est obtenu par application de la même formule *(cf. articles D. 832-10 et D. 842-6) du CCH :*

APL ou AL = K [L + C – L0]

- Bonus payment system of the French Army: Louvois/SourceSolde

Doubtful results



III - Les formules de calculs
DE L'APL ET DES AL

Secteur locatif ordinaire

Depuis la réforme intervenue le 1er janvier 2001, le montant de l'aide est obtenu par application de la même formule en AL et en APL *(cf. article D. 823-16 du CCH).*

APL ou AL = L+C-Pp

Logements-foyers

- Le montant de l'APL foyer est obtenu par application de la formule *(cf. article D. 832-24 du CCH) :*

APL = K [E – E0]

Avec application de deux barèmes, APL 1 foyer et APL 2 foyer *(cf. articles D. 832-25 et D. 832-26 du CCH)*

- Le montant de l'AL est obtenu selon la formule *(cf. article D. 842-15 du CCH) :*

AL = K [L + C – L0]

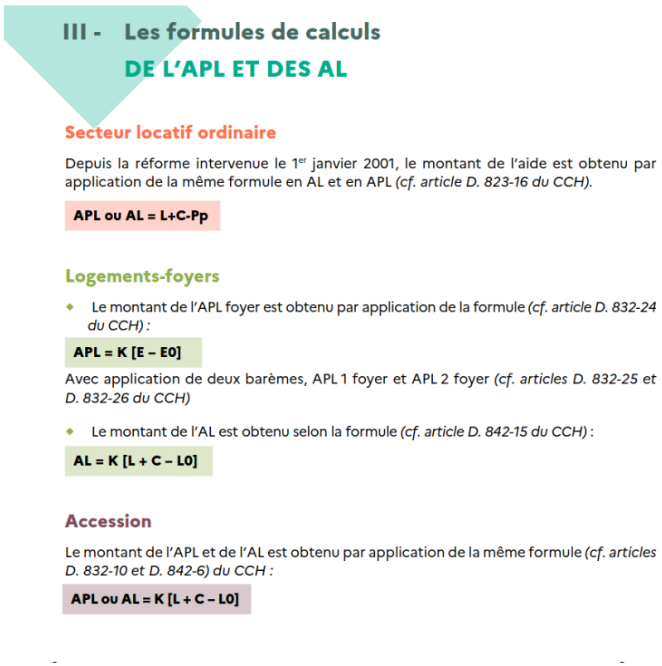
Accession

Le montant de l'APL et de l'AL est obtenu par application de la même formule *(cf. articles D. 832-10 et D. 842-6) du CCH :*

APL ou AL = K [L + C – L0]

- Bonus payment system of the French Army: Louvois/SourceSolde
- In 2012: 465 M € incorrect payments

Doubtful results



III - Les formules de calculs
DE L'APL ET DES AL

Secteur locatif ordinaire

Depuis la réforme intervenue le 1er janvier 2001, le montant de l'aide est obtenu par application de la même formule en AL et en APL *(cf. article D. 823-16 du CCH).*

APL ou AL = L+C-Pp

Logements-foyers

- Le montant de l'APL foyer est obtenu par application de la formule *(cf. article D. 832-24 du CCH) :*

APL = K [E – E0]

Avec application de deux barèmes, APL 1 foyer et APL 2 foyer *(cf. articles D. 832-25 et D. 832-26 du CCH)*

- Le montant de l'AL est obtenu selon la formule *(cf. article D. 842-15 du CCH) :*

AL = K [L + C – L0]

Accession

Le montant de l'APL et de l'AL est obtenu par application de la même formule *(cf. articles D. 832-10 et D. 842-6) du CCH :*

APL ou AL = K [L + C – L0]

- Bonus payment system of the French Army: Louvois/SourceSolde
- In 2012: 465 M € incorrect payments
- It left some soldiers and their families without any income at all for months!

Closer to (my) house: Civio vs Bosco

This article belongs to the debate » The Rule of Law versus the Rule of the Algorithm

02 April 2022

The Paradox of Efficiency: Frictions Between Law and Algorithms

On the 13th of January 2022, a Spanish Administrative court ruled in favour of algorithmic opacity. Fundación Civio, an independent foundation that monitors and accounts public authorities, reported that an algorithm used by the government was committing errors.[1)] BOSCO, the name of the application which contained the algorithm, was implemented by the Spanish public administration to more efficiently identify citizens eligible for grants to pay electricity bills. Meanwhile, Civio designed a web app to inform citizens whether they would be entitled for this grant.[2)] Thousands of citizens used this application and some of them reported that, while Civio's web app suggested

Ana Valdivia

Dr Ana Valdivia is a Postdoctoral Researcher at King's College London (ERC Security Flows). She examines how algorithms impact on people's life from a technical, political, and legal perspective.

Javier de la Cueva

Javier de la Cueva is a lawyer, lecturer and researcher in topics related to open knowledge, ethics and the digital world.

Explore posts related to this:
Algorithmic Efficiency, Algorithmic Justice, Rule of Law, Rule of the Algorithm

The Bosco computer program : errors in the computation of the social welfare bonuses

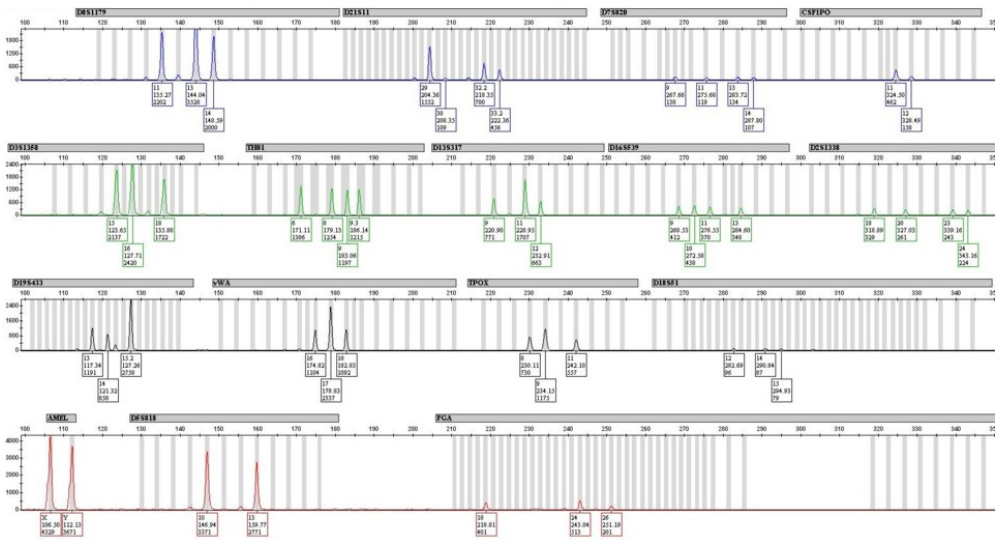
Least requirement: access to source code

In France it is mandatory to publish source code of software that is used in public administration.

However, access to source code will not resolve all problems

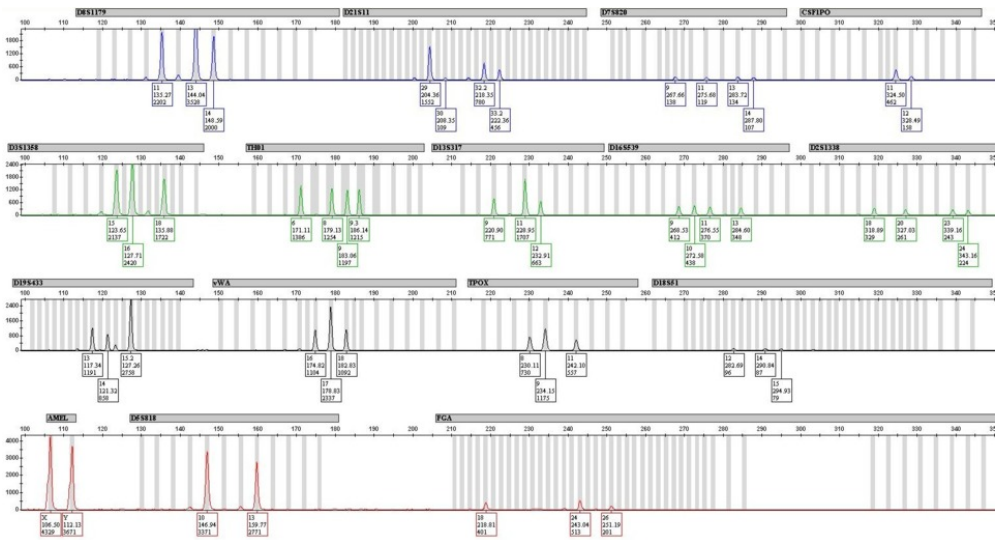
Evidencias dudosas

- In three law-suits in the USA, the defence requested to **open the proprietary source code to the jury** of DNA sequencing software since there were some doubts.
  - STRmix



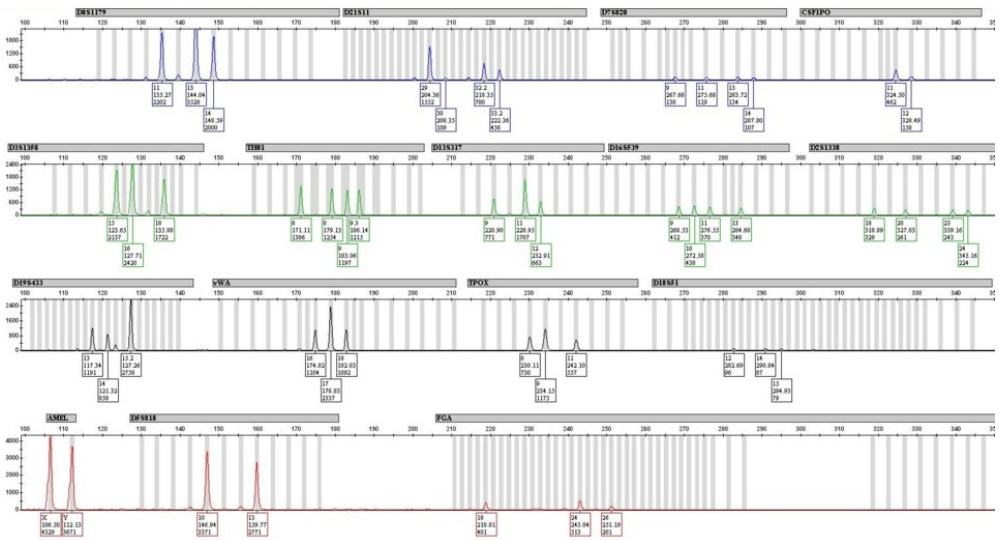
Evidencias dudosas

- In three law-suits in the USA, the defence requested to **open the proprietary source code to the jury** of DNA sequencing software since there were some doubts.
  - STRmix
  - FST



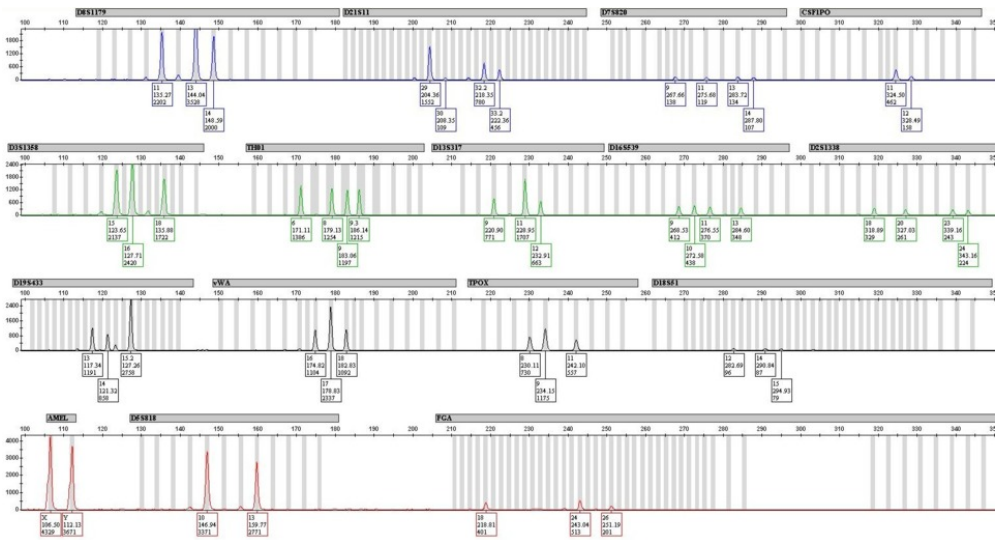
Evidencias dudosas

- In three law-suits in the USA, the defence requested to **open the proprietary source code to the jury** of DNA sequencing software since there were some doubts.
  - STRmix
  - FST
  - TrueAllele (not granted)



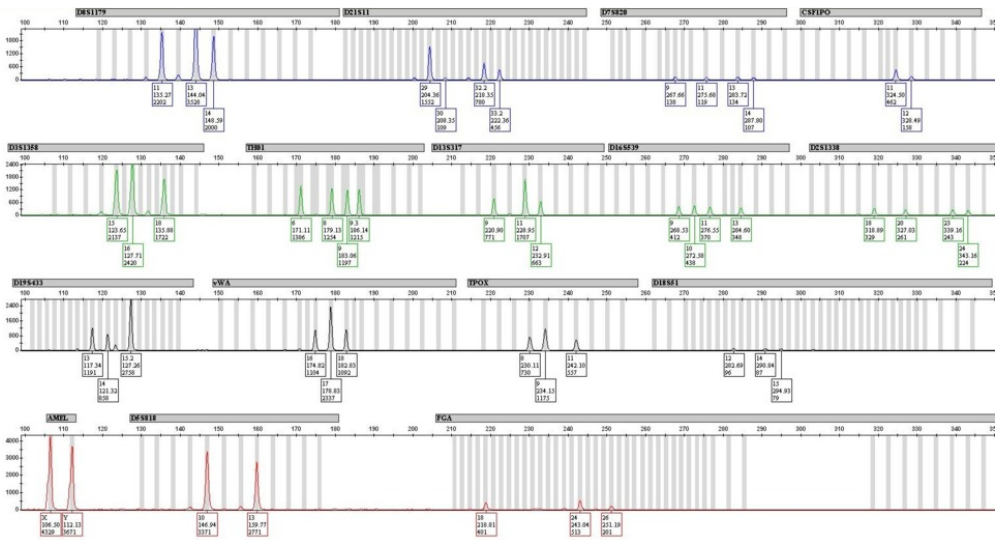
Evidencias dudosas

- In three law-suits in the USA, the defence requested to **open the proprietary source code to the jury** of DNA sequencing software since there were some doubts.
  - STRmix
  - FST
  - TrueAllele (not granted)
- In two cases the request was granted



Evidencias dudosas

- In three law-suits in the USA, the defence requested to **open the proprietary source code to the jury** of DNA sequencing software since there were some doubts.
  - STRmix
  - FST
  - TrueAllele (not granted)
- In two cases the request was granted
- Again, access to the source code will not solve all problems!



What is certification?

- Is it just a matter of trust? (combined with some sanity checks and experience)

What is certification?

- Is it just a matter of trust? (combined with some sanity checks and experience)
- Certificate $\Longrightarrow$ something is certain

What is certification?

- Is it just a matter of trust? (combined with some sanity checks and experience)
- Certificate $\Longrightarrow$ something is certain
- Verify $\Longrightarrow$ something is veridical

The impossibility of unrestricted certification

Alan Turing

- A mathematical theorem:

The impossibility of unrestricted certification

Alan Turing

- A mathematical theorem:
- Unrestricted certification is impossible.

Restricted certification is possible

We call a program P a *universal certifier* (wrt its language) when P takes two inputs

1. another program Q in a language compatible with P and,
2. a specification S in a language compatible with P that describes the behaviour of the program Q;

and, given two inputs Q and S, the program P outputs:

- **"YES"** if the program Q does what is said by S and, it will ouput
- **"NO"** if the program Q does something different as that what is claimed by S.

Theorem

There does not exist a universal certifier.
This holds for any reasonable class of languages.

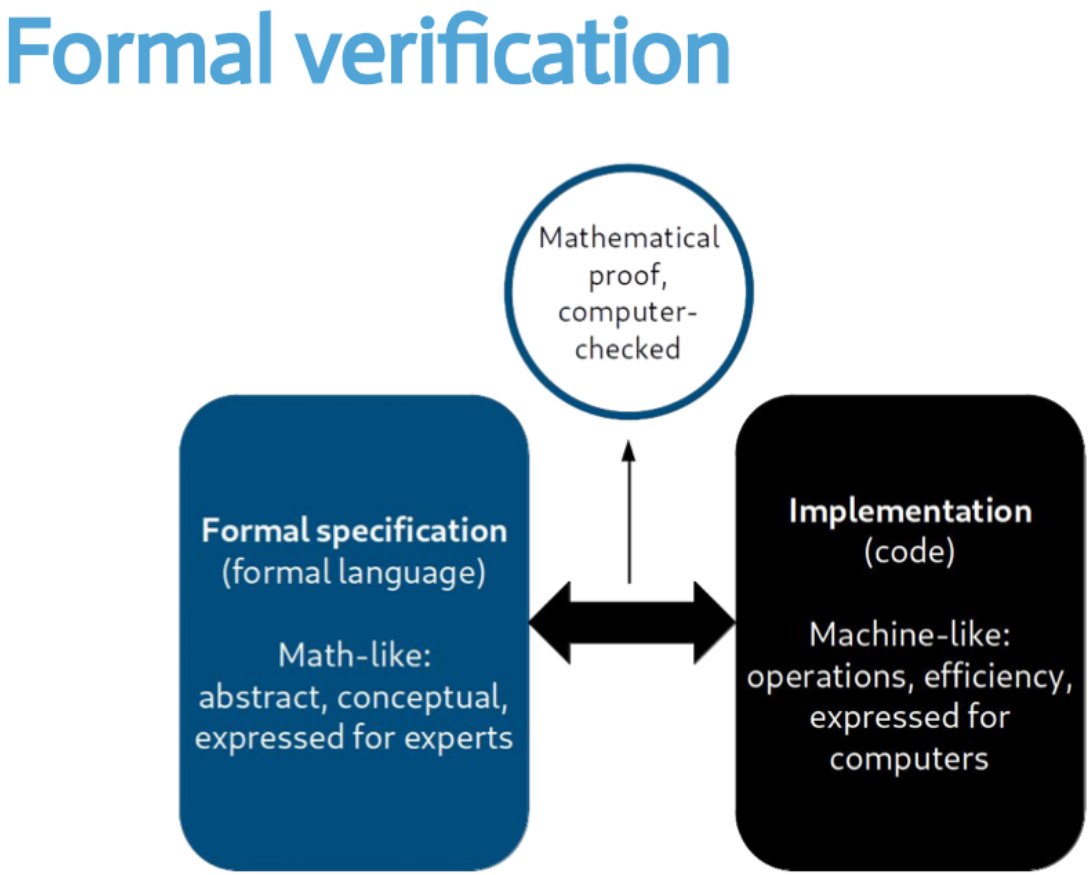
Formally verified software

Components of formally verified/certified software

- Σ A Specification: a non-ambiguous mathematical description of the input-output behaviour of the software
- Π Implementation: the code, the software, implementing the algorithm that does the work.
- Δ Proof: a mathematical proof that the program Π functions as claimed by Σ

The specification Σ is written in a formal language (in our case, the language of dependent types of the Coq proof assistant).
This begs the question: **How to make the specification more accessible to the general/judicial public?**

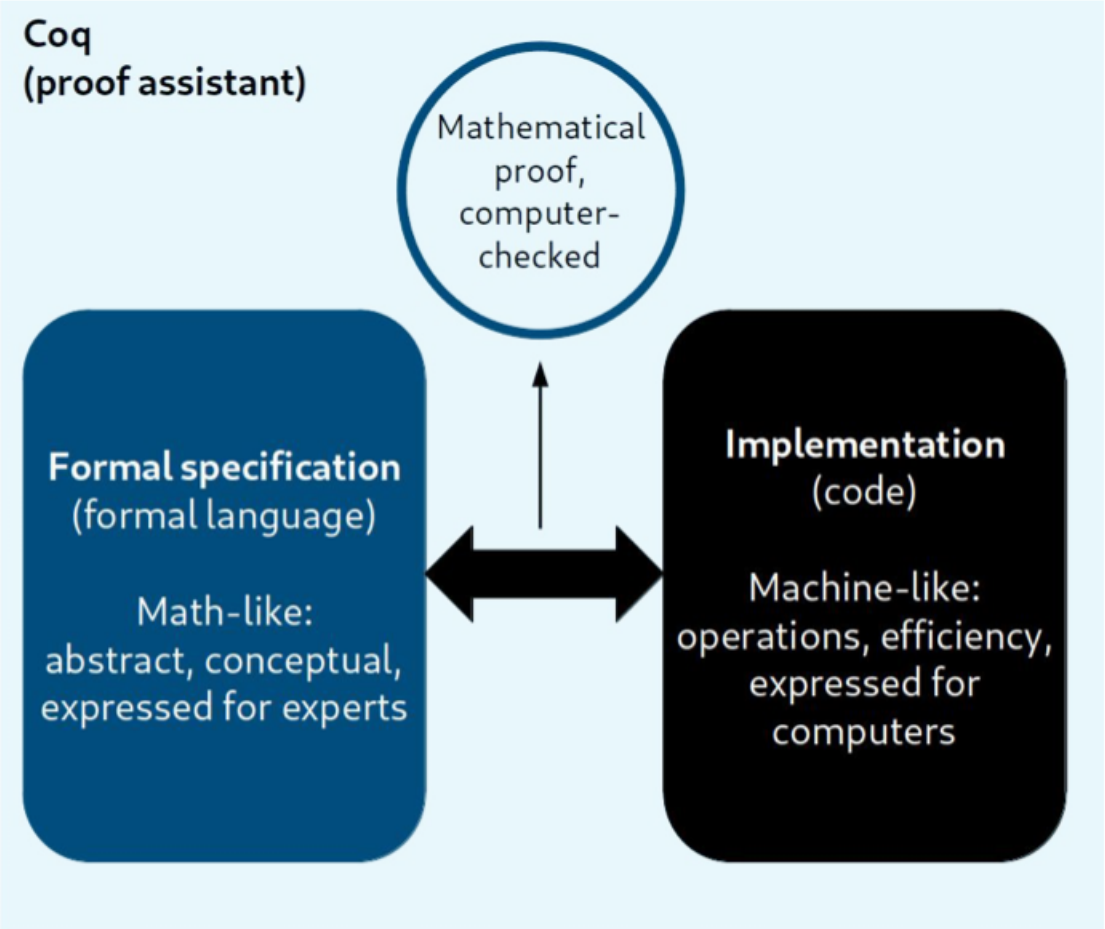
What is verification?



Slides FV: González Bedmar

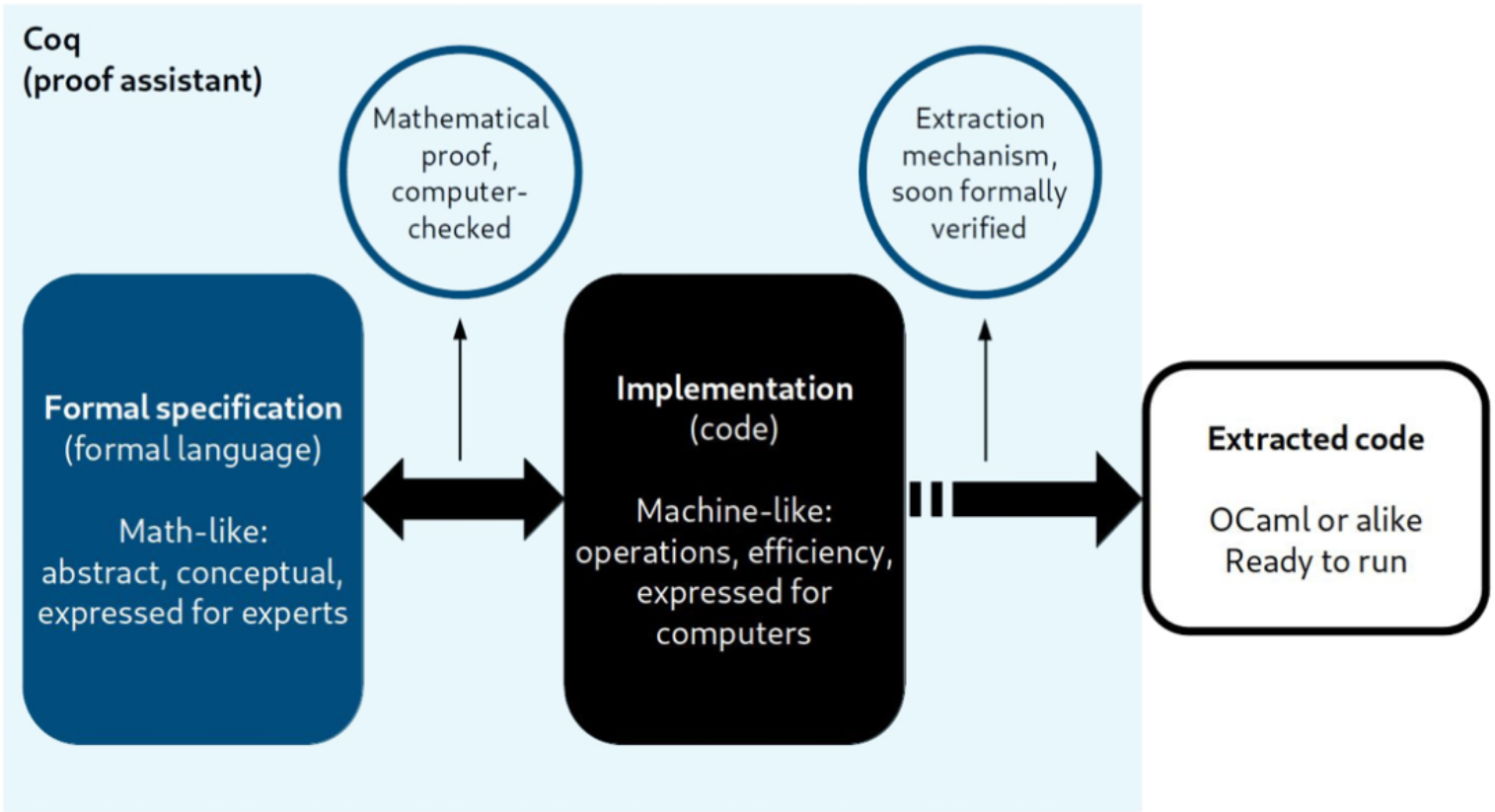
What is verification?

Formal verification

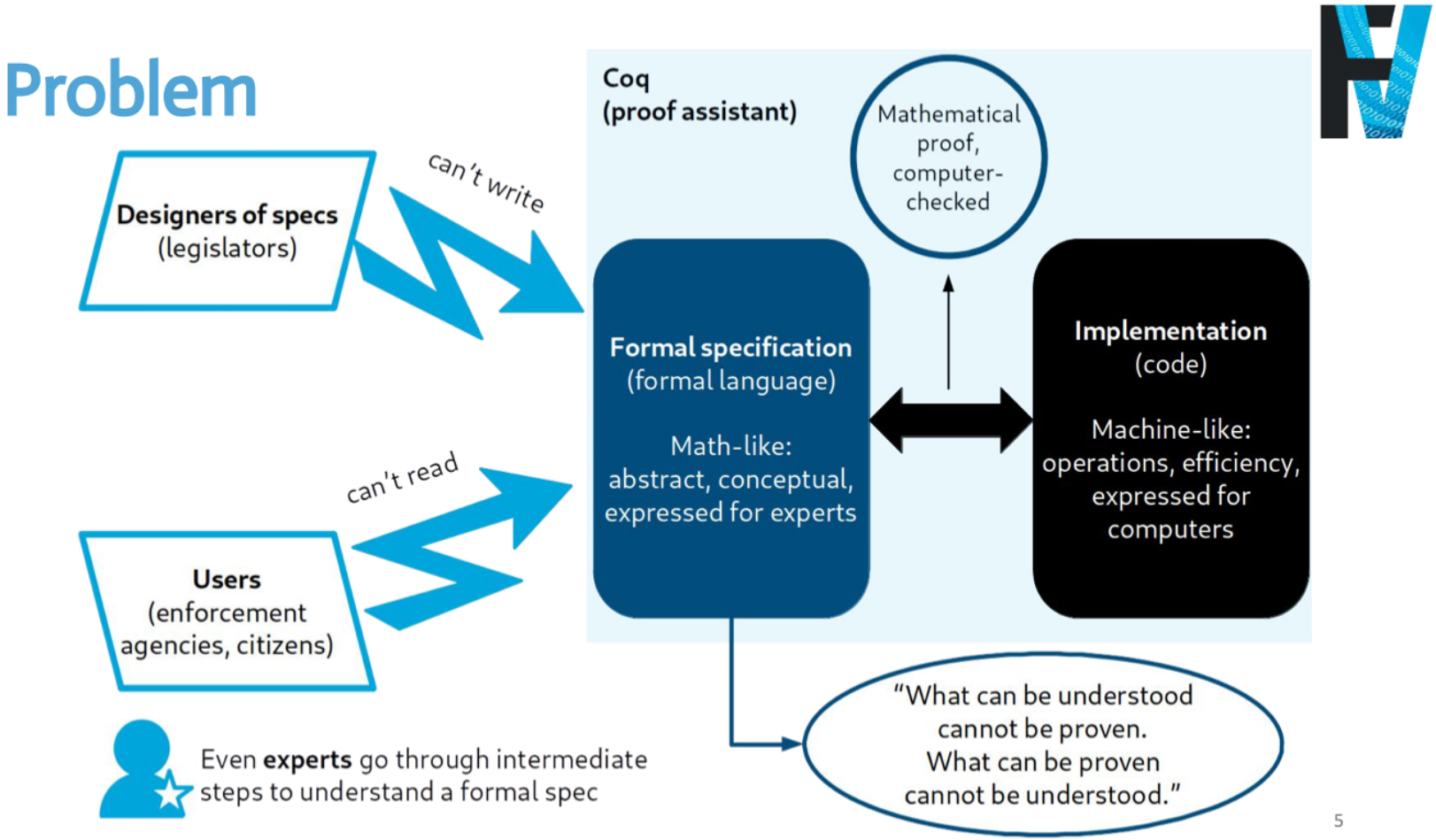


What is verification?

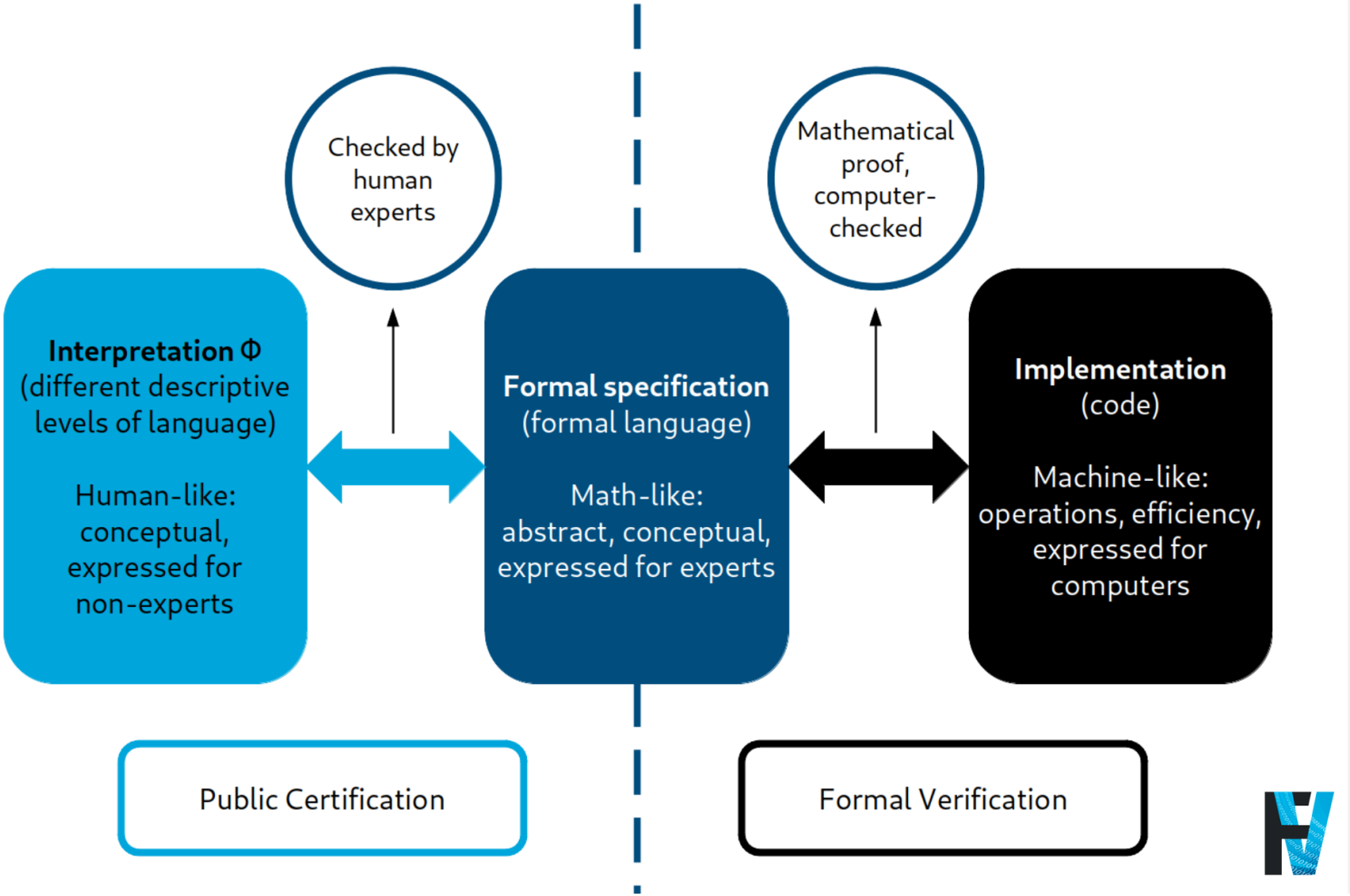
Formal verification



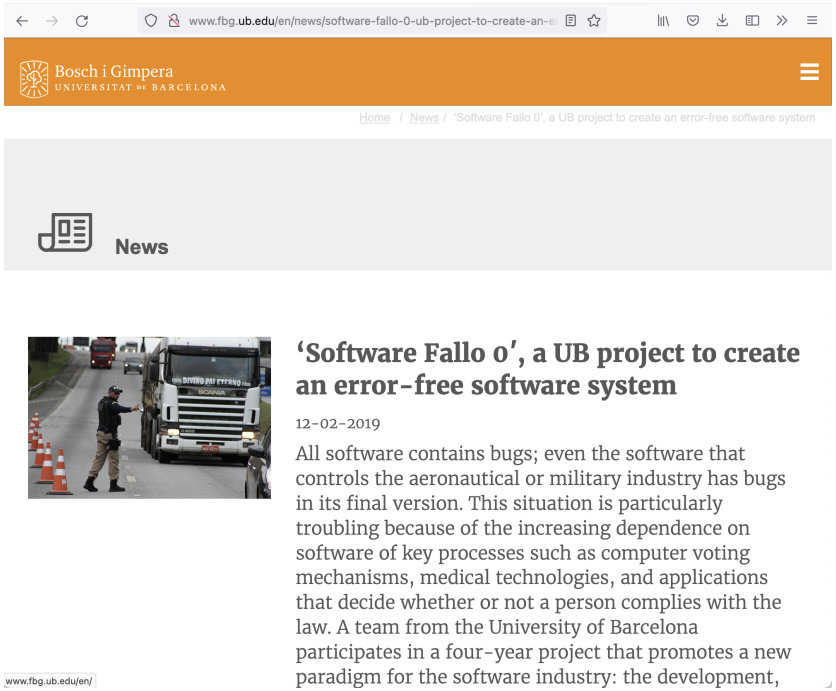
What is verification?

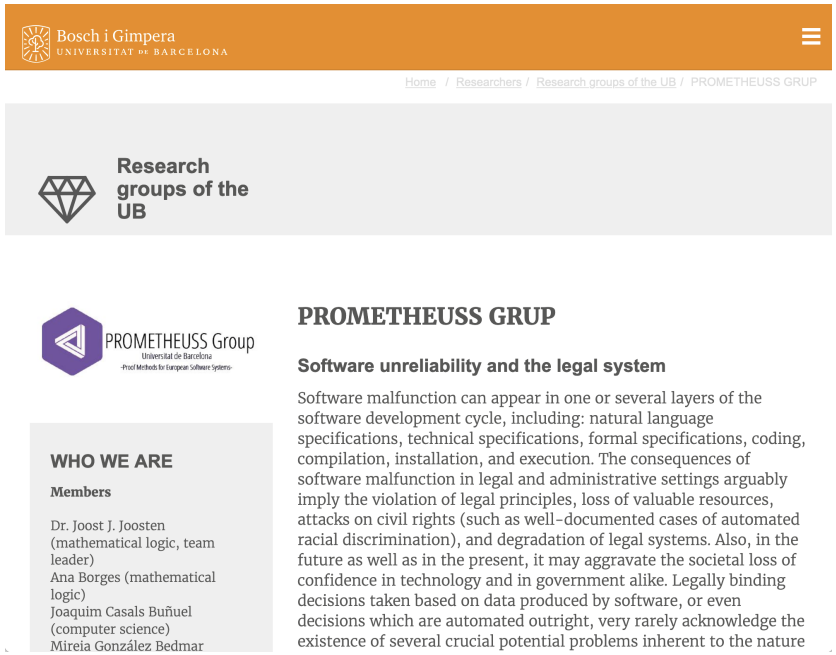


What is verification?



Seven years of research in Barcelona





Covenant between the University of Barcelona (FBG), Formal Vindications S.L. & Guretruck S.L.

Three weeks ago in Barcelona

Conference on Algorithmic Law Design and Implementation

APRIL 28-29, 2022

UNIVERSITAT DE BARCELONA

SCHEDULE, REGISTRATION and + INFO:

https://www.ub.edu/prooftheory/event/lawdesign/

KEYNOTE SPEAKERS

Formal Verification and Governance of Financial Algorithms with Imandra
Grant Olney Passmore
Imandra, USA

Hybrid intelligence for algorithmic law design
Bart Verheij
Bernoulli Institute of Mathematics, Computer Science and Artificial Intelligence · University of Groningen

INVITED SPEAKERS

Opening talk: **Algorithmic Law Design and Implementation. From Grave to Cradle**
Joost J. Joosten
Universitat de Barcelona, Spain

Logical Methods for Algorithmic Law
David Fernández-Duque
Ghent University, Belgium

Public Certification of Software and its necessity in Computable Laws. FV Time as the first application
Mireia González Bedmar
Formal Vindications SL, Spain

Is coding the law legal? A French and European approach
Liane Huttner
Université Paris 1 Panthéon-Sorbonne, France

Verifying well-behaved execution of legislative programs with the Catala domain-specific language
Denis Merigoux
Inria, Project Team Prosecco, France

Drafting EU Legislation in the Era of AI and Digitisation
Fernando Nubla Durango, Willy van Puymbroeck
LEOS Project, European Commission
Monica Palmirani
Università di Bologna, Italy

Auditing IT-systems used for automated individual decision-making in public sector; experiment in The Netherlands
Marlies van Eck
Hooghiemstra & Partners and Radboud University, The Netherlands

Verified extraction to OCaml from Coq, in Coq
Yannick Forster
Inria, Project Team Gallinette, France

Crafting a legislation ready for digital public administration
Julius Lyk-Jensen, Christine Holmgreen Mejling & Mette Eigaard Rasmussen
Agency for Digitalization, Ministry of Finance, Denmark

Model-Checking as an approach to algorithmic law and the case of Regulation 561
Moritz Müller
Universitat de Barcelona, Spain

Imagine lawyers are not your enemies: legal challenges and digital rights
Susana de la Sierra
Universidad de Castilla-La Mancha, Spain

PROGRAM COMMITTEE

Joost J. Joosten
Dept. of Philosophy, UB

Juli Ponce Solé
Dept. of Administrative Law, Procedural Law, History of Law, and Financial and Tax Law, UB

ORGANIZING COMMITTEE

Joost J. Joosten (Co-chair)
Dept. of Philosophy, UB

Juli Ponce Solé (Co-chair)
Dept. of Administrative Law, Procedural Law, History of Law, and Financial and Tax Law, UB

Petia Gaintchev Toneva (Secretary)
PhD candidate, Law Faculty, UB

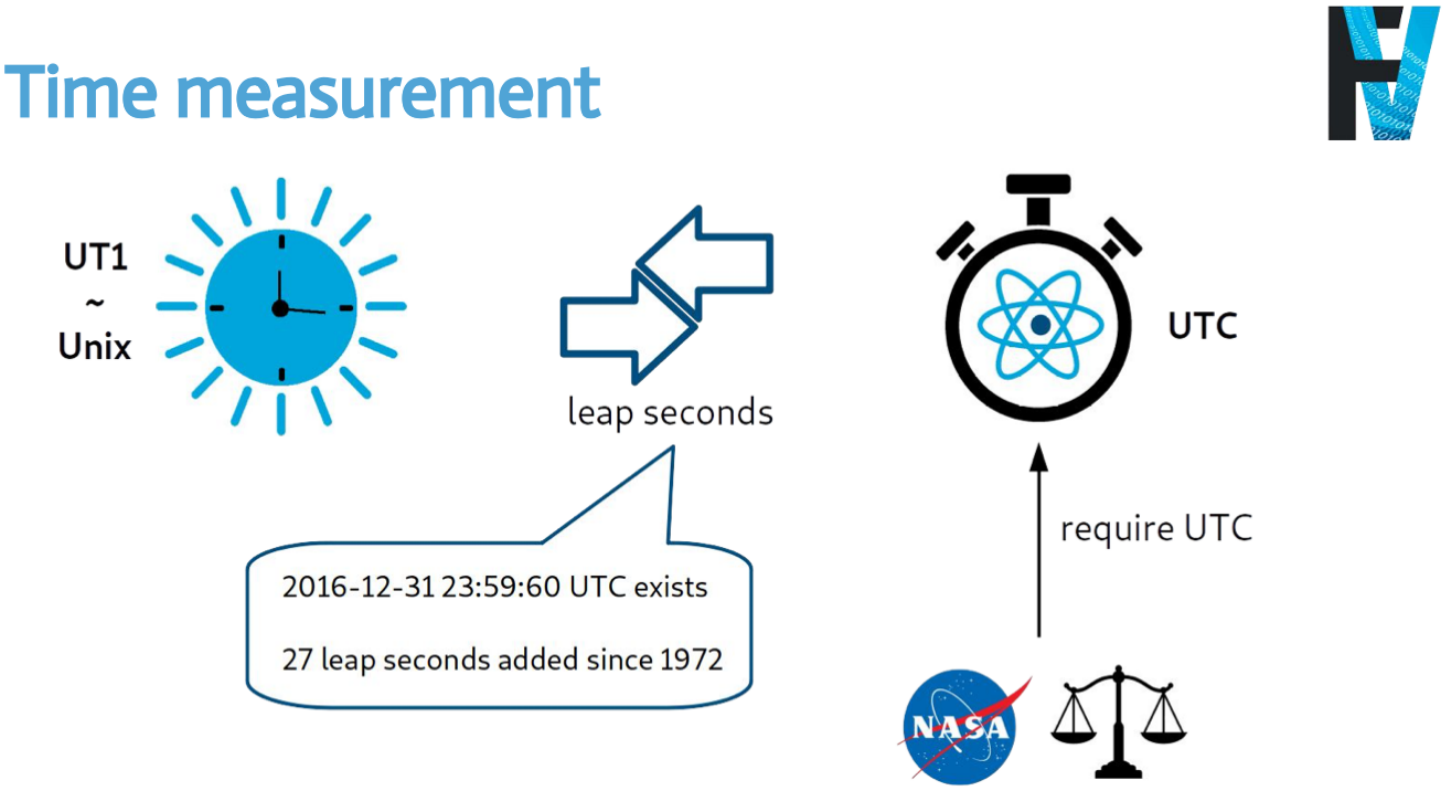
Gabriel Andrés Molero
Dept. of Philosophy, UB

Eric Sancho Adamson (poster design)
Dept. of Philosophy, UB

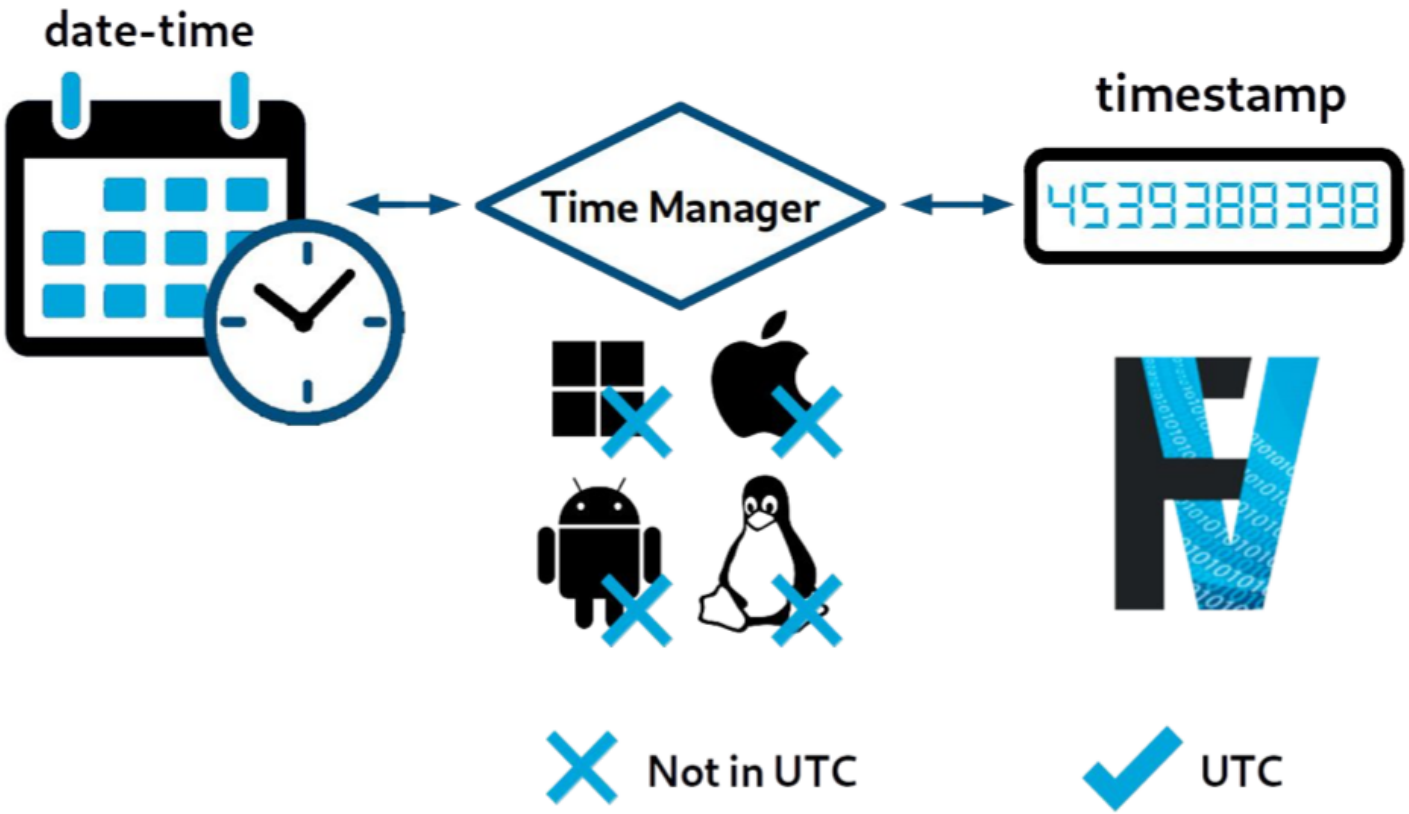
Isabel Serra Domènech
UB

Alex Solé Sánchez
Dept. of Philosophy, UB

Time library

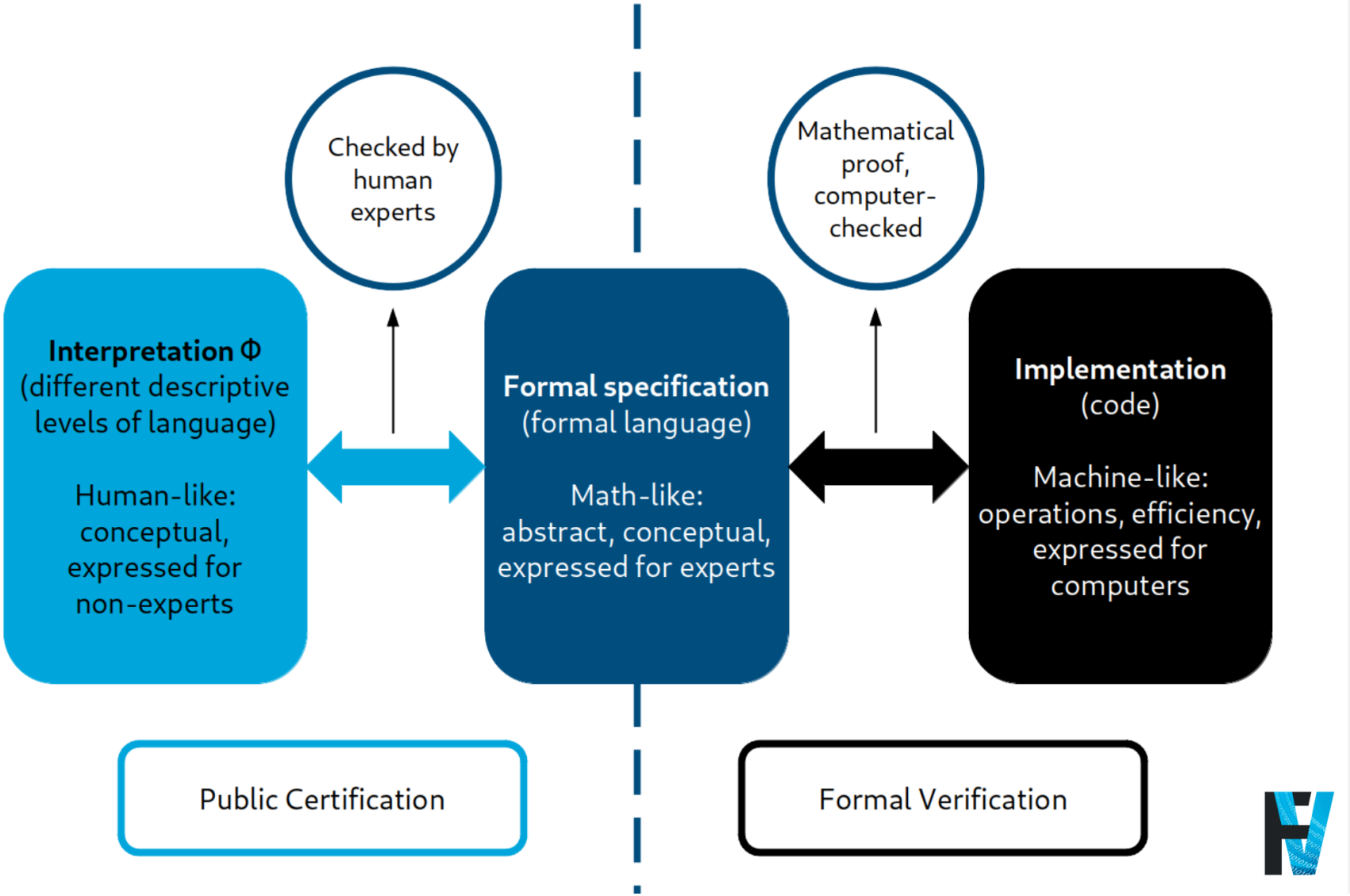


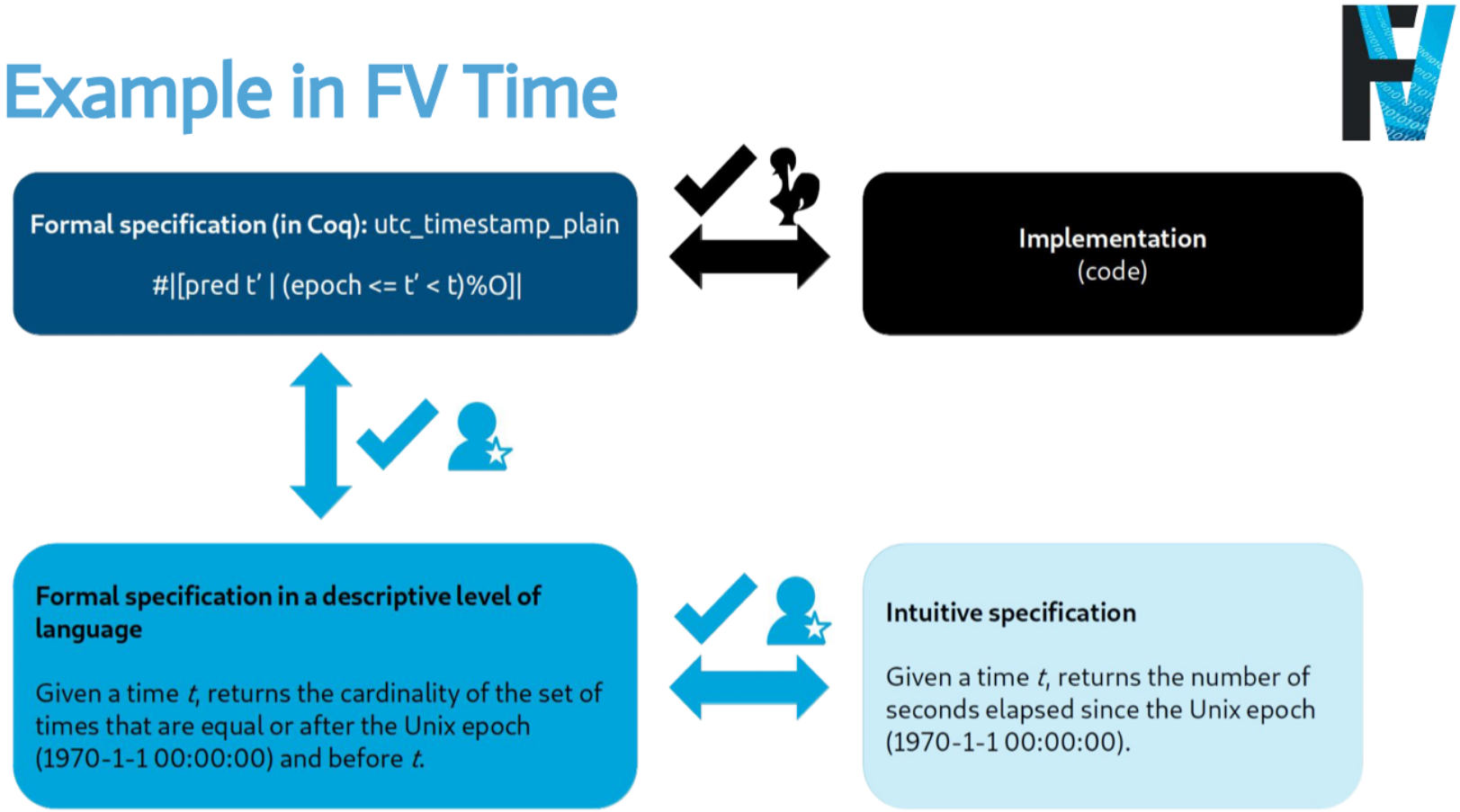
Time formats and managers



Most important feature: formally verified!

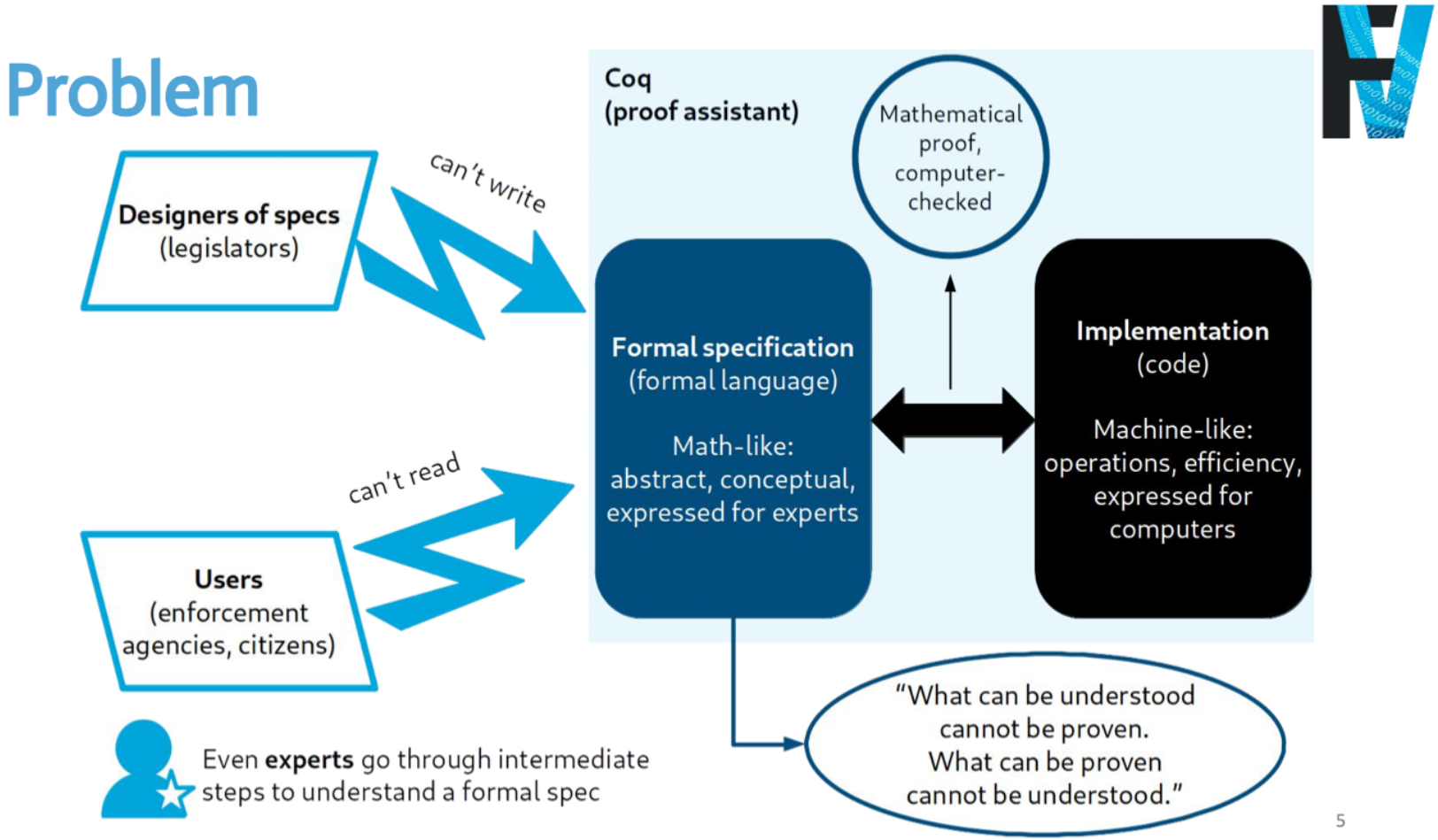
What is certification?



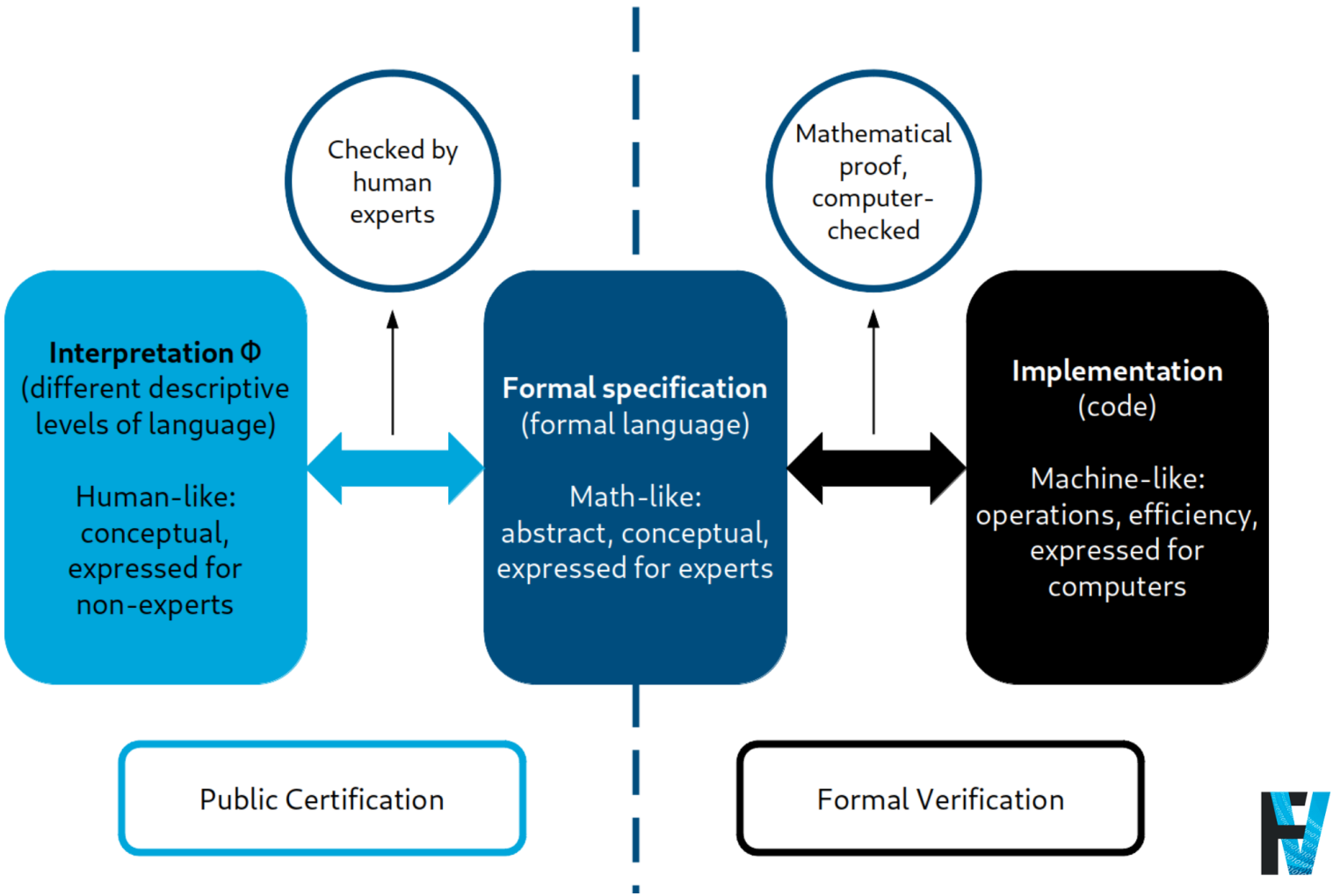


Around one-thousand times more expensive!

A central problem



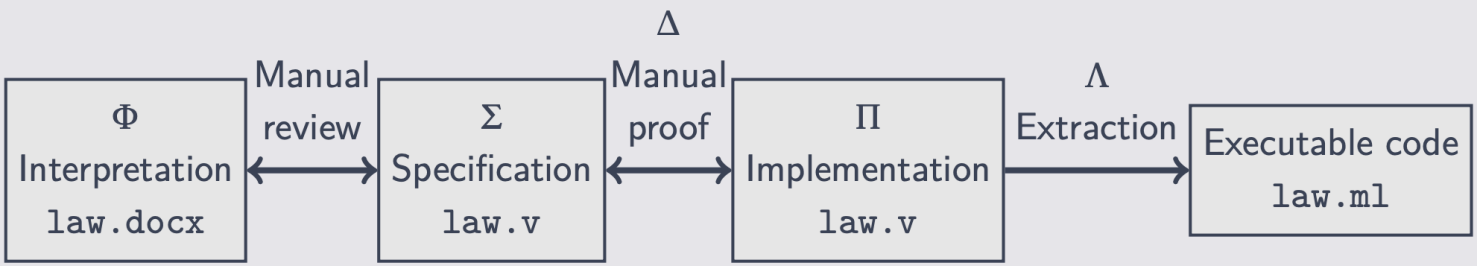
Public certification versus formal verification



Catala: A Shortcut For Legal Expert System Certification

The Usual Way to Produce Verified Software

Using Mireia Gonzáles Bedmar's conceptual framework from yesterday's presentation:



Catala's approach:

8

Slides Catala: Merigoux

❶ Catala: A Language Reviewable by lawyers

US Tax Code, Section 132, (c)(1) Qualified employee discount

The term "qualified employee discount" means any employee discount with respect to qualified property or services to the extent such discount does not exceed—
(A) in the case of property, the gross profit percentage of the price at which the property is being offered by the employer to customers

```
scope QualifiedEmployeeDiscount :
  definition qualified_employee_discount
    under condition is_property consequence equals
      if employee_discount >$ customer_price *$ gross_profit_percentage then
        customer_price *$ gross_profit_percentage
      else employee_discount
```

Can code be the law?

TENSION TABLE:

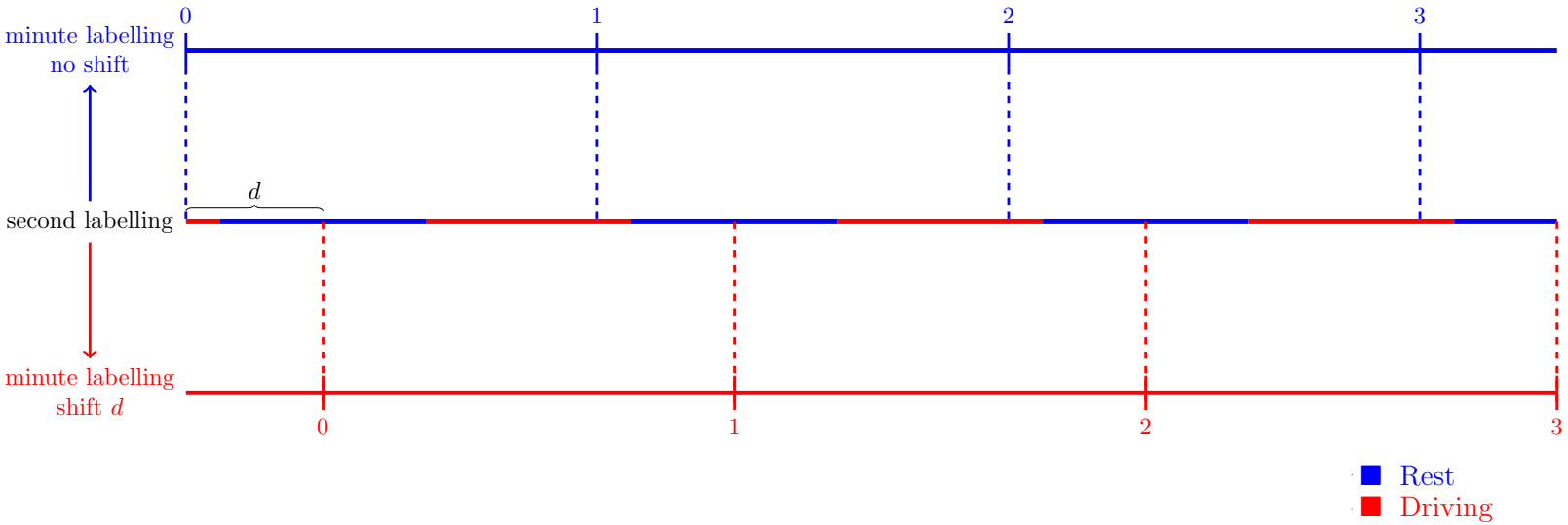
Computable laws:

Language, software paradigm and legal principles

Specification Language	Programming paradigm	Legal Certainty	Accountability	Contestability
Natural Language	Not Formally Verified	Decisions will probably not be consistent with the established legal framework. The text will be accessible and comprehensible to the public and authorities.	Automated decision won't be reliable and explainability will be difficult: the software is not comprehensible to the public, challenging the principle of transparency.	Right to contest turns almost impossible since authorities can't explain software decisions, which will be unreliable.
Technical Language	Not Formally Verified	Decisions will likely not be consistent with the established legal framework. The text will be less comprehensible to public and authorities.	Automated decision will be barely reliable and explainability will be difficult: the software is not comprehensible to the public, challenging the principle of transparency.	Right to contest turns almost impossible since authorities can't explain software decisions, which will be mostly unreliable.
Formal Language	Not Formally Verified	Decisions will probably be consistent with the established legal framework. The text will only be accessible to experts.	Automated decision will be quite reliable and explainability will be difficult: the software is not comprehensible to the public, challenging the principle of transparency.	Right to contest turns almost impossible since authorities can't explain software decisions, yet they will probably be working according to the law
Formal Language	Formally Verified	Decisions will be consistent with established legal framework. The text will only be accessible to experts	Automated decision will be reliable and explainability will be difficult, but it will be guaranteed that the software is the exact reproduction of its specification	Right to contest will be difficult since authorities can't explain software decisions, yet those are working according to the law

More accurate and exact but less understandable for the general public

Further benefits of formalisation



We proved that the labelling is not *shift-invariant*!

Some regulations regarding weekly rest periods

Regulation (EC) No 561/2006

§8.6. In any two consecutive weeks, a driver shall take at least:

- two regular weekly rest periods [of at least 45 hours], or
- one regular weekly rest period and one reduced weekly rest period of at least 24 hours. However, the reduction shall be compensated by an equivalent period of rest taken en bloc before the end of the third week following the week in question.

A weekly rest period shall start no later than at the end of six 24-hour periods from the end of the previous weekly rest period.

§8.9. A weekly rest period that falls in two weeks may be counted in either week, but not in both.

Let's break it down...

- Regular weekly rest: $\geq$ 45 hours

Let's break it down...

- Regular weekly rest: $\geq$ 45 hours
- Reduced weekly rest: $\geq$ 24 hours

Let's break it down...

- Regular weekly rest: $\geq$ 45 hours
- Reduced weekly rest: $\geq$ 24 hours
- Each rest period is assigned to only one week it intersects

Let's break it down...

- Regular weekly rest: $\geq$ 45 hours
- Reduced weekly rest: $\geq$ 24 hours
- Each rest period is assigned to only one week it intersects
- Every week must have a regular or reduced weekly rest

Let's break it down...

- Regular weekly rest: $\geq$ 45 hours
- Reduced weekly rest: $\geq$ 24 hours
- Each rest period is assigned to only one week it intersects
- Every week must have a regular or reduced weekly rest
- Every other week must have a full weekly rest

Let's break it down...

- Regular weekly rest: $\geq$ 45 hours
- Reduced weekly rest: $\geq$ 24 hours
- Each rest period is assigned to only one week it intersects
- Every week must have a regular or reduced weekly rest
- Every other week must have a full weekly rest
- Any reduced rest must be compensated by a continuous block in the following three weeks

Combinatorics of rest assignments

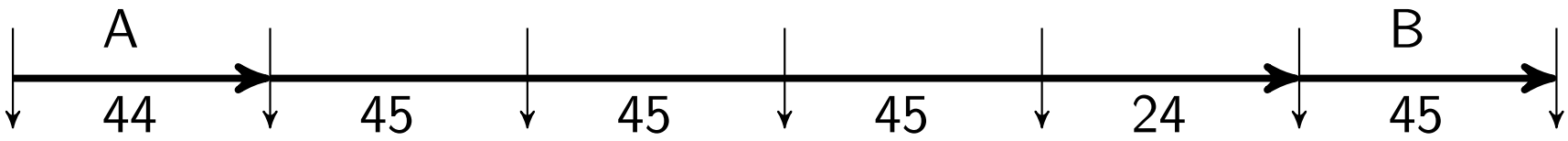
Can we assign a week to each rest period so that each week is assigned to at least one rest period?

Combinatorics of rest assignments

Can we assign a week to each rest period so that each week is assigned to at least one rest period?

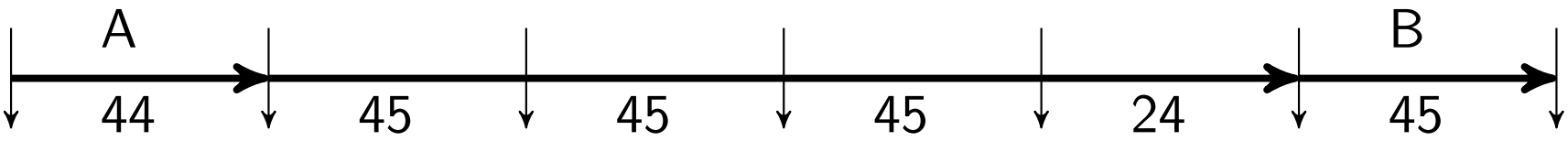
In principle this is an NP problem (assign 0 or 1 to each rest period according to whether it should belong to the earlier week or the later week).

Non-locality of compensations

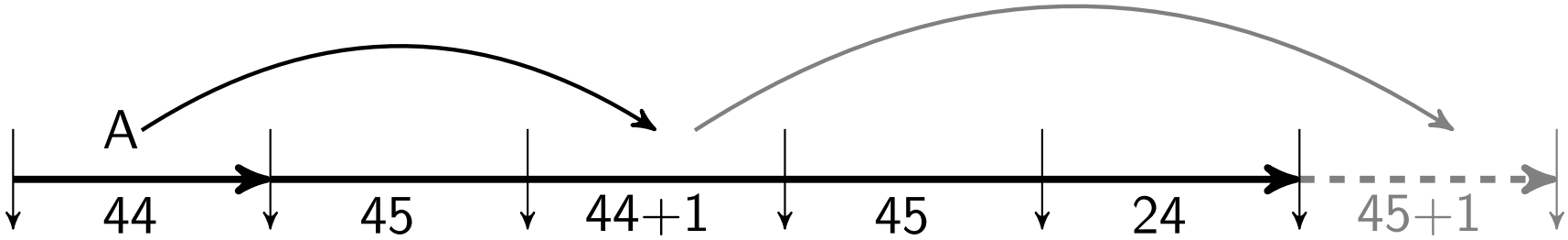


Illegal

Non-locality of compensations

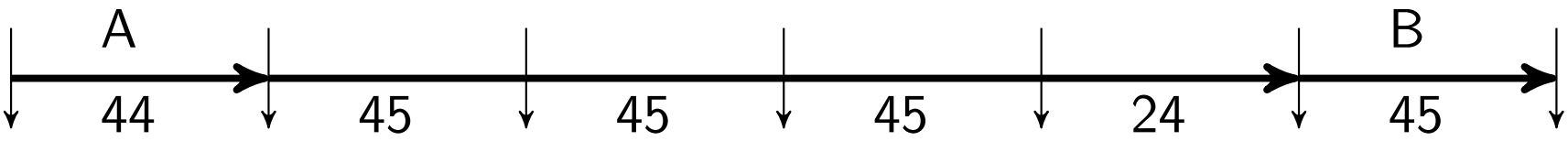


Illegal

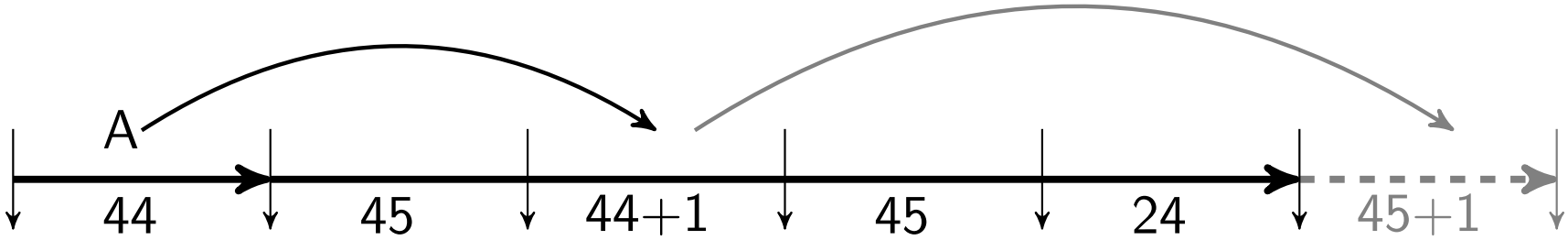


Legal

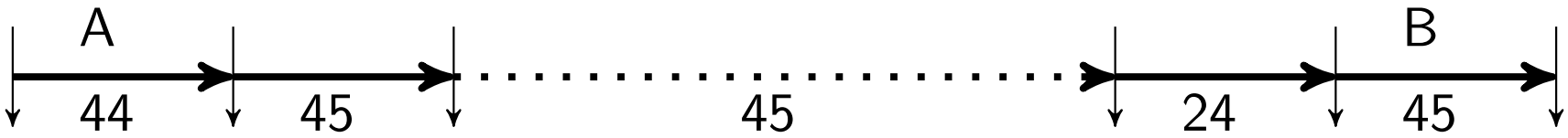
Non-locality of compensations



Illegal



Legal



This can be iterated indefinitely: non-locality

Lab activities

- Regulation analysis via logical/mathematical analysis

Lab activities

- Regulation analysis via logical/mathematical analysis
  - Is the regulation consistent?

Lab activities

- Regulation analysis via logical/mathematical analysis
  - Is the regulation consistent?
  - Is the implied behaviour of the regulation desirable?

Lab activities

- Regulation analysis via logical/mathematical analysis
  - Is the regulation consistent?
  - Is the implied behaviour of the regulation desirable?
    - Shift-invariance of labelling, locality of legality checks, etc.

Lab activities

- Regulation analysis via logical/mathematical analysis
  - Is the regulation consistent?
  - Is the implied behaviour of the regulation desirable?
    - Shift-invariance of labelling, locality of legality checks, etc.
    - But also: if x and y are similar data containers (e.g., formal repr. of persons) with the only difference that x has one passport and y has two passports, will the program behave the same for x and y?

Lab activities

- Regulation analysis via logical/mathematical analysis
  - Is the regulation consistent?
  - Is the implied behaviour of the regulation desirable?
    - Shift-invariance of labelling, locality of legality checks, etc.
    - But also: if x and y are similar data containers (e.g., formal repr. of persons) with the only difference that x has one passport and y has two passports, will the program behave the same for x and y?
  - Are the algorithms implied computationally feasible?

Lab activities

- Regulation analysis via logical/mathematical analysis
  - Is the regulation consistent?
  - Is the implied behaviour of the regulation desirable?
    - Shift-invariance of labelling, locality of legality checks, etc.
    - But also: if x and y are similar data containers (e.g., formal repr. of persons) with the only difference that x has one passport and y has two passports, will the program behave the same for x and y?
  - Are the algorithms implied computationally feasible?
- Implement software in a ZERO-ERROR fashion using proof assistants.

Lab activities

- Regulation analysis via logical/mathematical analysis
  - Is the regulation consistent?
  - Is the implied behaviour of the regulation desirable?
    - Shift-invariance of labelling, locality of legality checks, etc.
    - But also: if x and y are similar data containers (e.g., formal repr. of persons) with the only difference that x has one passport and y has two passports, will the program behave the same for x and y?
  - Are the algorithms implied computationally feasible?
- Implement software in a ZERO-ERROR fashion using proof assistants.
- Develop general purpose models with the above considerations taken into account so that zero-error software scales.

Lab activities

- Regulation analysis via logical/mathematical analysis
  - Is the regulation consistent?
  - Is the implied behaviour of the regulation desirable?
    - Shift-invariance of labelling, locality of legality checks, etc.
    - But also: if x and y are similar data containers (e.g., formal repr. of persons) with the only difference that x has one passport and y has two passports, will the program behave the same for x and y?
  - Are the algorithms implied computationally feasible?
- Implement software in a ZERO-ERROR fashion using proof assistants.
- Develop general purpose models with the above considerations taken into account so that zero-error software scales.
- Provide verified software with a dialogue fragment that enables a possible rudimentary dialogue between the user and the software about the software's behaviour

Lab activities

- Develop explanatory certificates by

Lab activities

- Develop explanatory certificates by
  - choosing ontologies of the right granularity to define semi-formal language

Lab activities

- Develop explanatory certificates by
  - choosing ontologies of the right granularity to define semi-formal language
  - conversion of lambda-terms to ones using in these ontologies/constructors

Lab activities

- Develop explanatory certificates by
  - choosing ontologies of the right granularity to define semi-formal language
  - conversion of lambda-terms to ones using in these ontologies/constructors
- Zero-knowledge certificates for proprietary software!

Lab activities

- Develop explanatory certificates by
  - choosing ontologies of the right granularity to define semi-formal language
  - conversion of lambda-terms to ones using in these ontologies/constructors
- Zero-knowledge certificates for proprietary software!
- Standards and good practices for public certification.

Lab activities

- Develop explanatory certificates by
  - choosing ontologies of the right granularity to define semi-formal language
  - conversion of lambda-terms to ones using in these ontologies/constructors
- Zero-knowledge certificates for proprietary software!
- Standards and good practices for public certification.
- A formal subset of natural language (domain specific) to bridge the gap between formal specifications and technical specifications (public certification)

Lab activities

- Develop explanatory certificates by
  - choosing ontologies of the right granularity to define semi-formal language
  - conversion of lambda-terms to ones using in these ontologies/constructors
- Zero-knowledge certificates for proprietary software!
- Standards and good practices for public certification.
- A formal subset of natural language (domain specific) to bridge the gap between formal specifications and technical specifications (public certification)
- Teaching courses on formal verification/certification techniques (and consulting)

Lab activities

- Develop explanatory certificates by
  - choosing ontologies of the right granularity to define semi-formal language
  - conversion of lambda-terms to ones using in these ontologies/constructors
- Zero-knowledge certificates for proprietary software!
- Standards and good practices for public certification.
- A formal subset of natural language (domain specific) to bridge the gap between formal specifications and technical specifications (public certification)
- Teaching courses on formal verification/certification techniques (and consulting)
- Adding to Coq development and libraries

Lab activities

- Develop explanatory certificates by
  - choosing ontologies of the right granularity to define semi-formal language
  - conversion of lambda-terms to ones using in these ontologies/constructors
- Zero-knowledge certificates for proprietary software!
- Standards and good practices for public certification.
- A formal subset of natural language (domain specific) to bridge the gap between formal specifications and technical specifications (public certification)
- Teaching courses on formal verification/certification techniques (and consulting)
- Adding to Coq development and libraries
- Study impact of techniques on society and ethical/legal principles, e.g., Accessibility, Accountability, Transparency, Equality, Contestability, etc

Lab activities

- Develop explanatory certificates by
  - choosing ontologies of the right granularity to define semi-formal language
  - conversion of lambda-terms to ones using in these ontologies/constructors
- Zero-knowledge certificates for proprietary software!
- Standards and good practices for public certification.
- A formal subset of natural language (domain specific) to bridge the gap between formal specifications and technical specifications (public certification)
- Teaching courses on formal verification/certification techniques (and consulting)
- Adding to Coq development and libraries
- Study impact of techniques on society and ethical/legal principles, e.g., Accessibility, Accountability, Transparency, Equality, Contestability, etc

The central computational problem of algorithmic law

(Work and slides with Moritz Müller)
Need to formalize activity sequences and laws

- formalize activity sequences are words $w \in \Sigma^*$ over a finite alphabet Σ
  e.g, dddrrw formalizes 6 minutes of activities in $\Sigma = \{d, r, w\}$.
- formalize a law by a sentence in a suitable logic L.

Need algorithm that decides the computational problem

$MC(\Sigma^*, L)$
Input: a word $w \in \Sigma^*$ *and a sentence* $\varphi \in L$
Problem: is w *legal according to* φ*, i.e.* $w \models \varphi$ *?*

$\mathrm{MC}(\Sigma^*, L)$ is a formal model for algorithmic law (on activity sequences).
Question For which L is it good?

Candidate: monadic second order logic MSO

Starting point

Borges, Conejero, Fernández-Duque, González, Joosten.
To drive or not to drive: A logical and computational analysis of European transport regulations. Information and Computation 280, 2021.

- naturally formalizes Regulation 561.
- model-checking in time $f(|\varphi|) \cdot |w|$, Parameterized Complexity
  where $f : \mathbb{N} \to \mathbb{N}$ is some computable function.
- but f grows very fast:

  Theorem *(Frick, Grohe 04)*
  Assume $P \neq NP$. Then MC(Σ^*,MSO) is not decidable in time

$$f(|\varphi|) \cdot |w|^{O(1)}$$

  for elementary $f : \mathbb{N} \to \mathbb{N}$.

Hence MSO is not sufficiently tractable.

Candidate: linear time temporal logic LTL

Model-checking in time $O(|\varphi| \cdot |w|)$, but not sufficiently expressive and not sufficiently succinct (BRFB21)

Example `Article 6.2: The weekly driving time shall not exceed 56 hours`

Straightforwardly formalized over words of length $1w$: disjunction of

$$\bigwedge_{d \leq D} \left(\bigwedge_{r_d \leq i < \ell_{d+1}} \bigcirc^i \neg d \wedge \bigwedge_{\ell_d \leq i < r_d} \bigcirc^i d \right)$$

for all $D \leq 1w$ and

all $r_0 := 0 \leq \ell_1 < r_1 < \cdots < \ell_D < r_D < \ell_{D+1} := 1w$ with

$$\sum_{1 \leq j \leq D} (r_j - \ell_j) \leq 56h$$

This has $> \binom{7 \cdot 24 \cdot 60}{56 \cdot 60} > 10^{2784}$ many disjuncts.

Warning

Algorithmic laws could use large constants for time constraints.

Model-checking complexity should scale well with them.

Which MC(Σ^*, L) are good models for algorithmic law?

Tractability

sufficiently fast model-checkers

fine-grained complexity analysis: parameterized complexity theory

important parameter: large time constants in law

Expressivity

test case: formalize Regulation 561

Naturality

readable sentences

sufficiently succinct

Stopwatch automata SWA: syntax

Stopwatch automaton $\mathbb{A}$

- Q *finite set of states including start, accept*
- X *finite set of stopwatches*
- λ *maps* $q \in Q$ *to* $\lambda(q) \in \Sigma$
- β *maps* $x \in X$ *to bound* $\beta(x) \in \mathbb{N}$
- ζ *is the set of* $(x, q) \in X \times Q$ *such that* x *is active in* q
- Δ *is the set of transitions* (q, g, α, q')
  where $q, q' \in Q$, g *is a guard,* α *is an action.*

Assignment ξ maps $x \in X$ to $\xi(x) \leq \beta(x)$

Guard g is a set of assignments

Action α maps assignments to assignments

Stopwatch automata SWA: semantics

Transition system of $\mathbb{A}$

configurations (q, ξ)

switch edges $(q, \xi) \xrightarrow{0} (q', \xi')$

whenever $(q, g, \alpha, q') \in \Delta$, $\xi \in g$, $\xi' = \alpha(\xi)$

stay edges $(q, \xi) \xrightarrow{t} (q, \xi')$

where ξ' *increases* $\xi(x)$ *for* x *active in* q *to* $\min\{\xi(x)+t, \beta(x)\}$

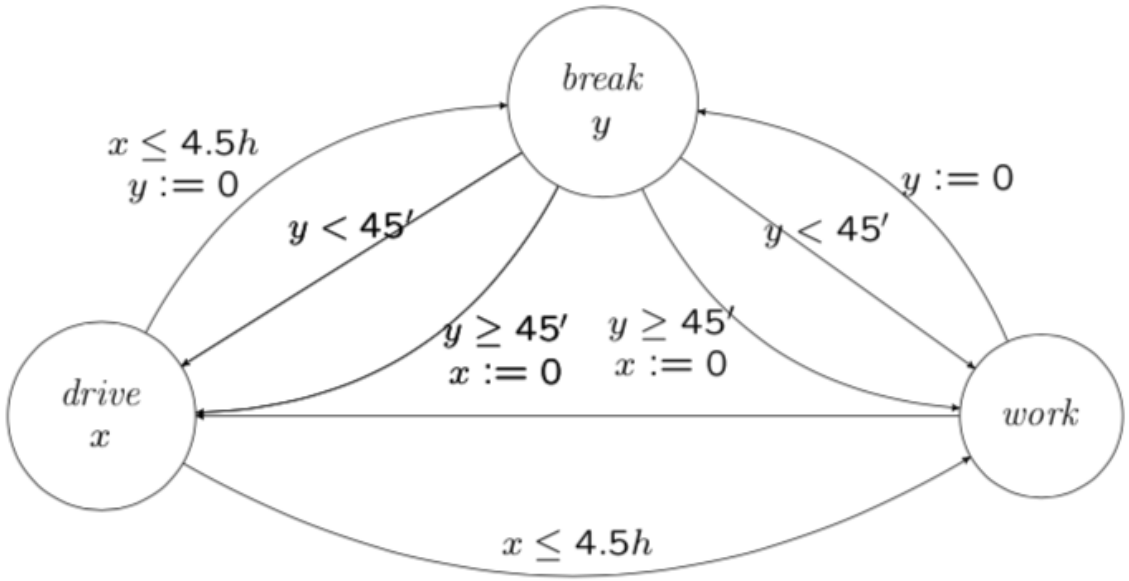
Computation $(q_0, \xi_0) \xrightarrow{t_0} (q_1, \xi_1) \xrightarrow{t_1} (q_2, \xi_2) \xrightarrow{t_2} \cdots \xrightarrow{t_{\ell-1}} (q_\ell, \xi_\ell)$

reads $w := \lambda(q_0)^{t_0}\ \lambda(q_1)^{t_1} \cdots \lambda(q_{\ell-1})^{t_{\ell-1}}$

accepts if $q_0 = \mathit{start}$, $\xi_0 \equiv 0$, $q_\ell = \mathit{accept}$, $q_i \neq \mathit{accept}$ for $i < \ell$.

Example: continuous driving

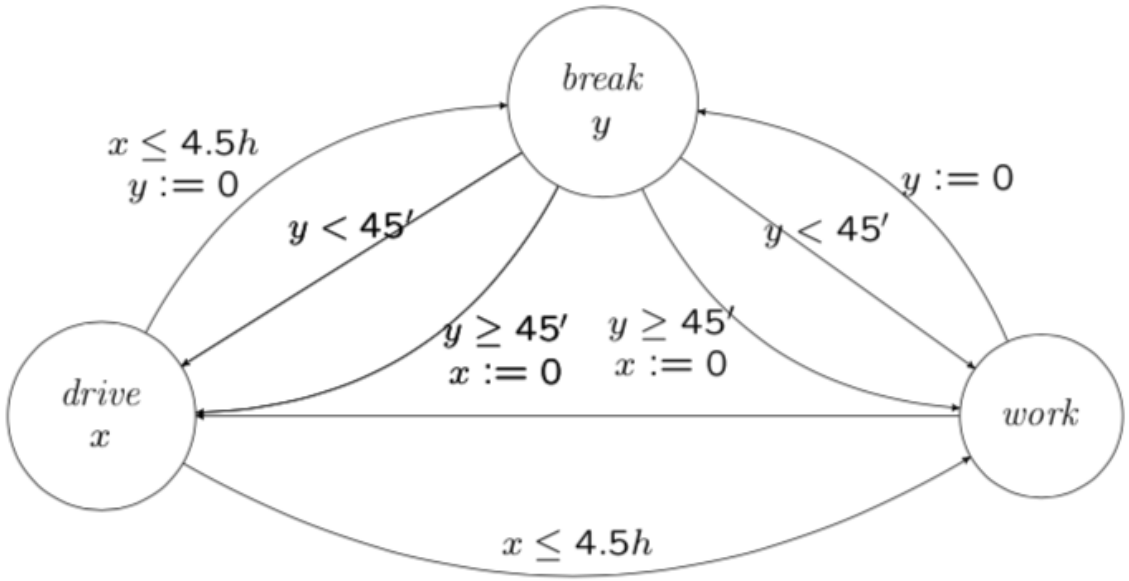
`Article 7 (1st part): After a driving period of four and a half hours a driver shall take an uninterrupted break of not less than 45 minutes,...`



$(drive, 00) \xrightarrow{3} (drive, 30) \xrightarrow{0} (break, 30) \xrightarrow{2} (break, 32) \xrightarrow{0} (work, 32) \xrightarrow{0} (break, 30)$
reads *dddrr*

Example: continuous driving

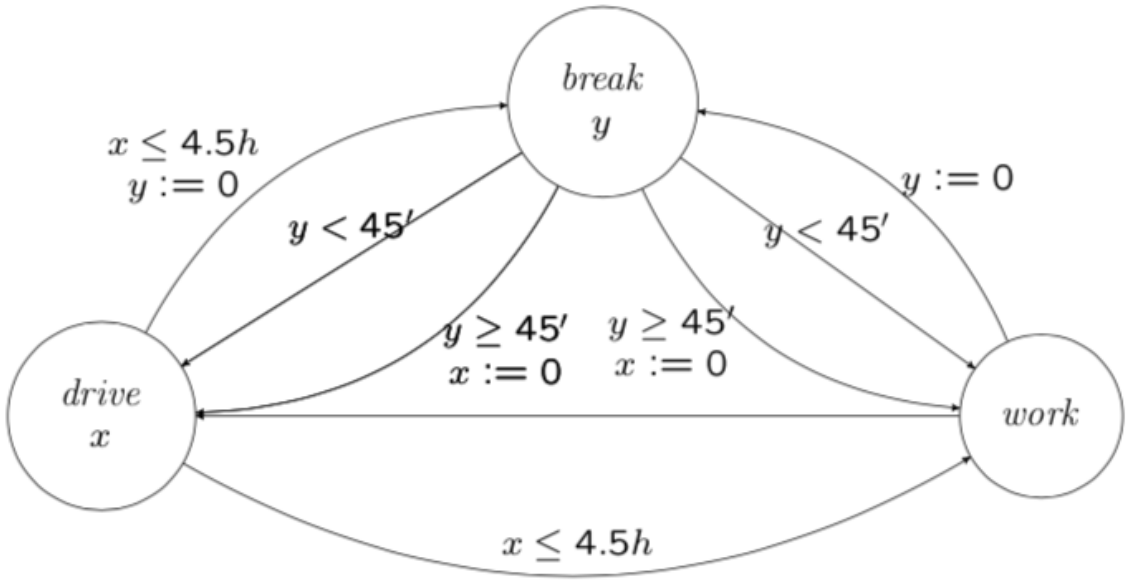
Article 7 (1st part): After a driving period of four and a half hours a driver shall take an uninterrupted break of not less than 45 minutes,...



$(drive, 00) \xrightarrow{3} (drive, 30) \xrightarrow{0} (break, 30) \xrightarrow{2} (break, 32) \xrightarrow{0} (work, 32) \xrightarrow{0} (break, 30)$
reads *dddrr*

Example: continuous driving

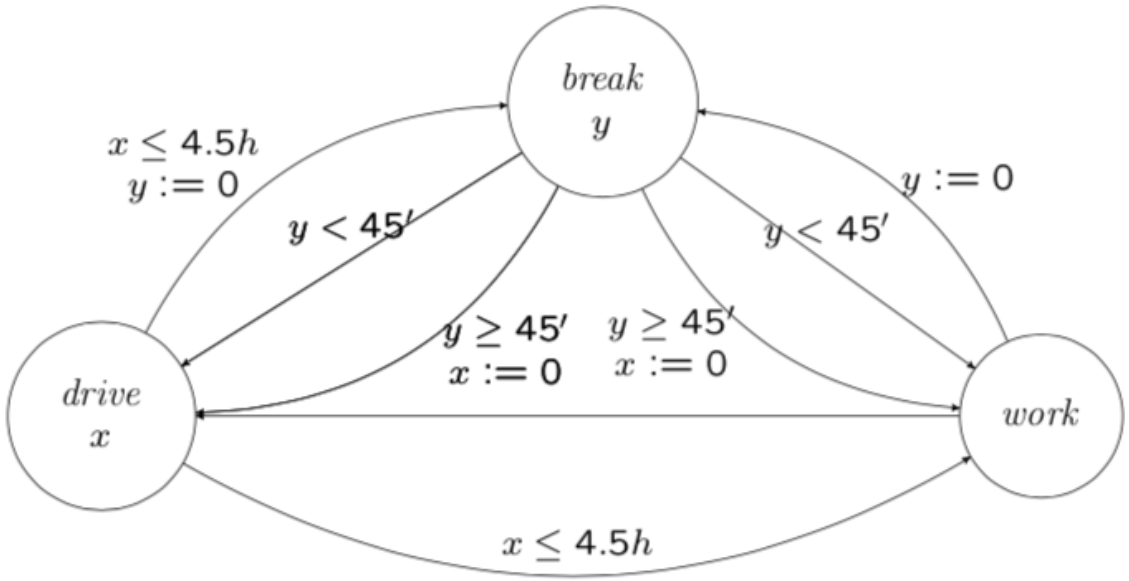
Article 7 (1st part): After a driving period of four and a half hours a driver shall take an uninterrupted break of not less than 45 minutes,...



$(drive, 00) \xrightarrow{3} (drive, 30) \xrightarrow{0} (break, 30) \xrightarrow{2} (break, 32) \xrightarrow{0} (work, 32) \xrightarrow{0} (break, 30)$
reads *dddrr*

Example: continuous driving

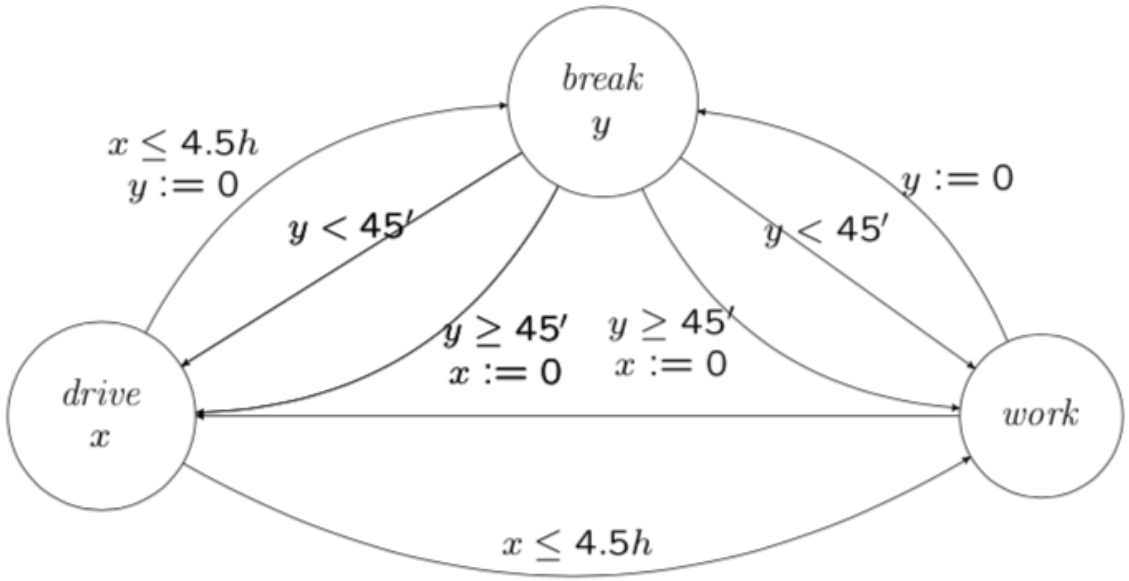
Article 7 (1st part): After a driving period of four and a half hours a driver shall take an uninterrupted break of not less than 45 minutes,...



$(drive, 00) \xrightarrow{3} (drive, 30) \xrightarrow{0} (break, 30) \xrightarrow{2} (break, 32) \xrightarrow{0} (work, 32) \xrightarrow{0} (break, 30)$
reads *dddrr*

Example: continuous driving

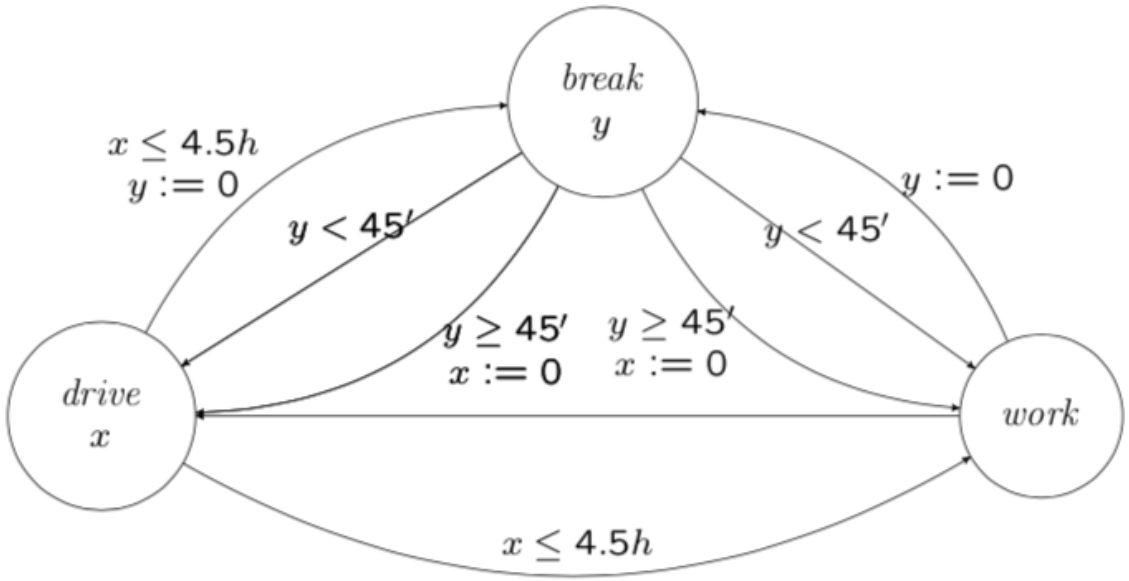
Article 7 (1st part): After a driving period of four and a half hours a driver shall take an uninterrupted break of not less than 45 minutes,...



$(drive, 00) \xrightarrow{3} (drive, 30) \xrightarrow{0} (break, 30) \xrightarrow{2} (break, 32) \xrightarrow{0} (work, 32) \xrightarrow{0} (break, 30)$
reads *dddrr*

Example: continuous driving

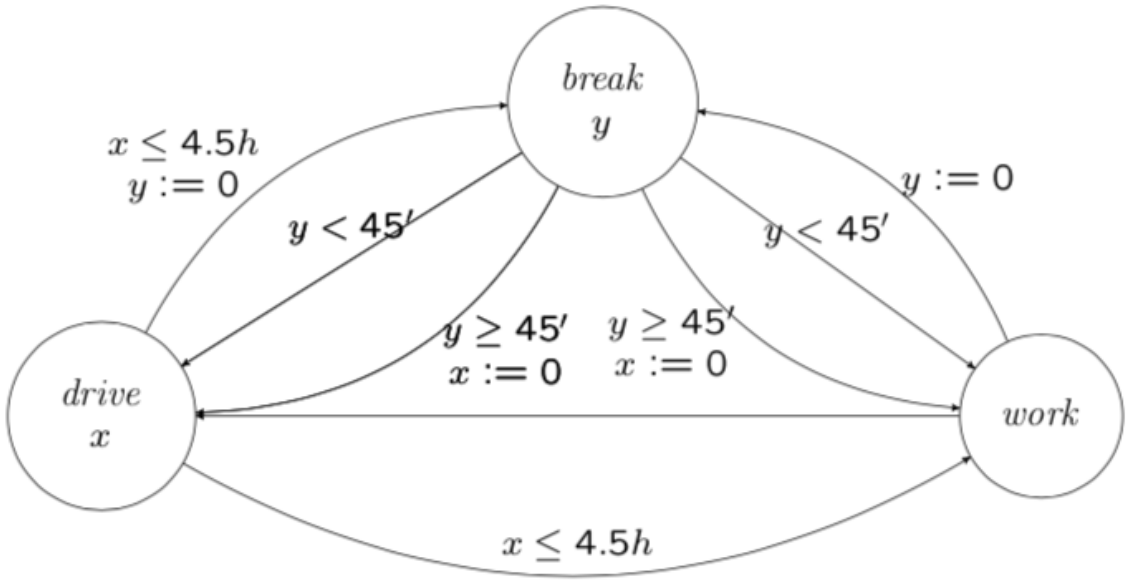
Article 7 (1st part): After a driving period of four and a half hours a driver shall take an uninterrupted break of not less than 45 minutes,...



$(drive, 00) \xrightarrow{3} (drive, 30) \xrightarrow{0} (break, 30) \xrightarrow{2} (break, 32) \xrightarrow{0} (work, 32) \xrightarrow{0} (break, 30)$
reads *dddrr*

Example: continuous driving

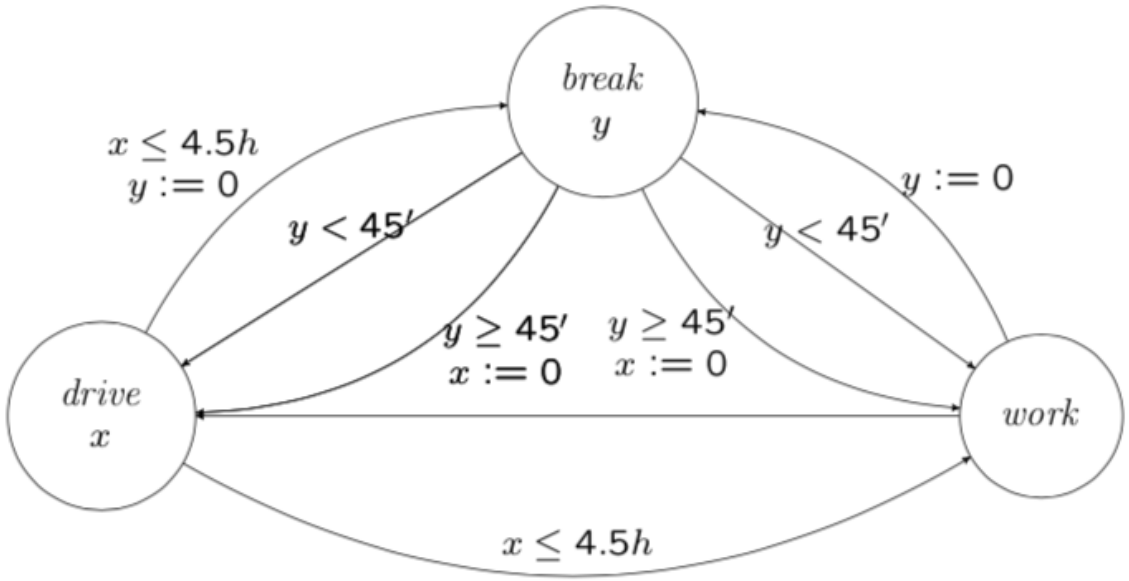
Article 7 (1st part): After a driving period of four and a half hours a driver shall take an uninterrupted break of not less than 45 minutes,...



$(drive, 00) \xrightarrow{3} (drive, 30) \xrightarrow{0} (break, 30) \xrightarrow{2} (break, 32) \xrightarrow{0} (work, 32) \xrightarrow{0}$
$(break, 30)$
reads *dddrr*

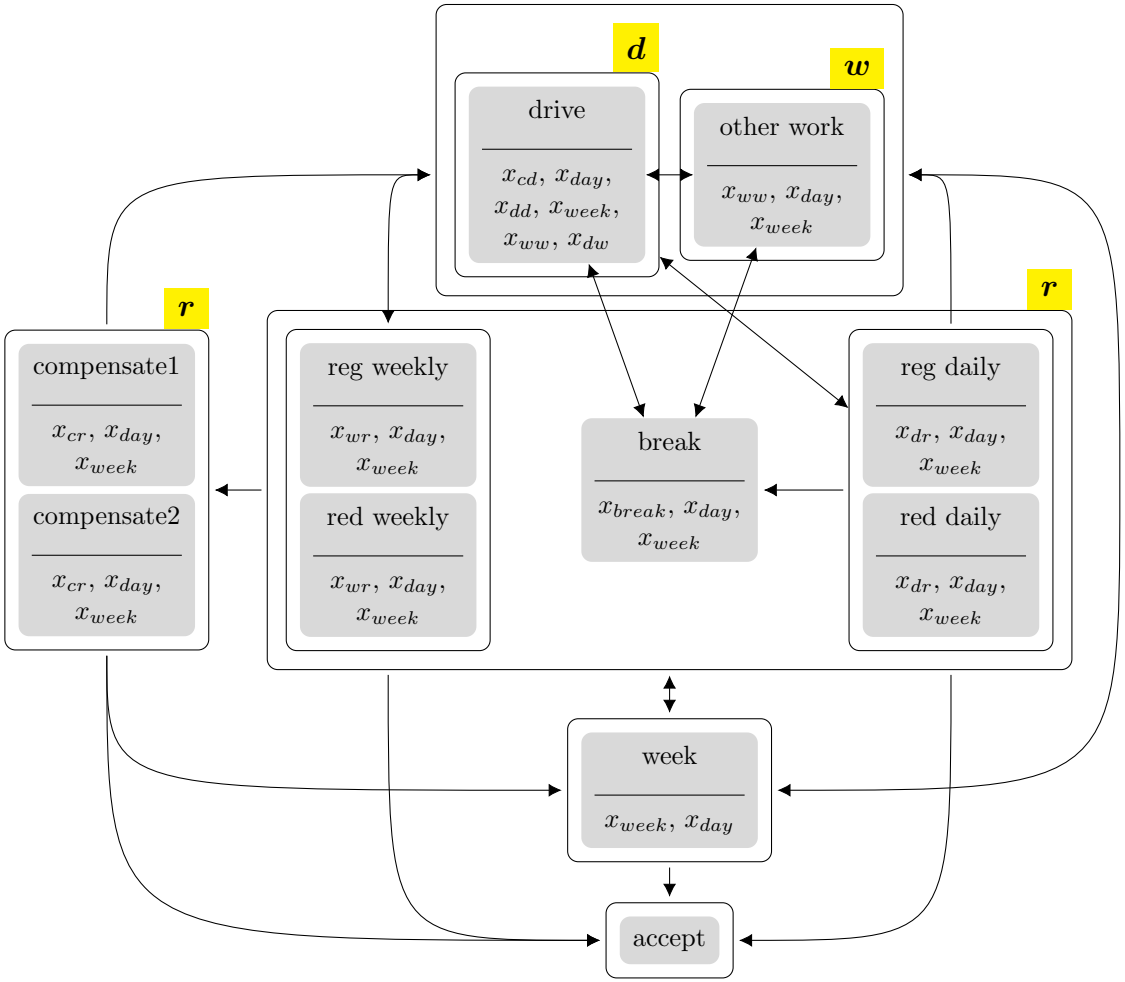
Example: continuous driving

Article 7 (1st part): After a driving period of four and a half hours a driver shall take an uninterrupted break of not less than 45 minutes,...



$(drive, 00) \xrightarrow{3} (drive, 30) \xrightarrow{0} (break, 30) \xrightarrow{2} (break, 32) \xrightarrow{0} (work, 32) \xrightarrow{0} (break, 30)$
reads *ddd rr*

Automaton that accepts exactly the legal words according to Reg. 561



12 states

$>$ 100 transitions

34 stopwatches

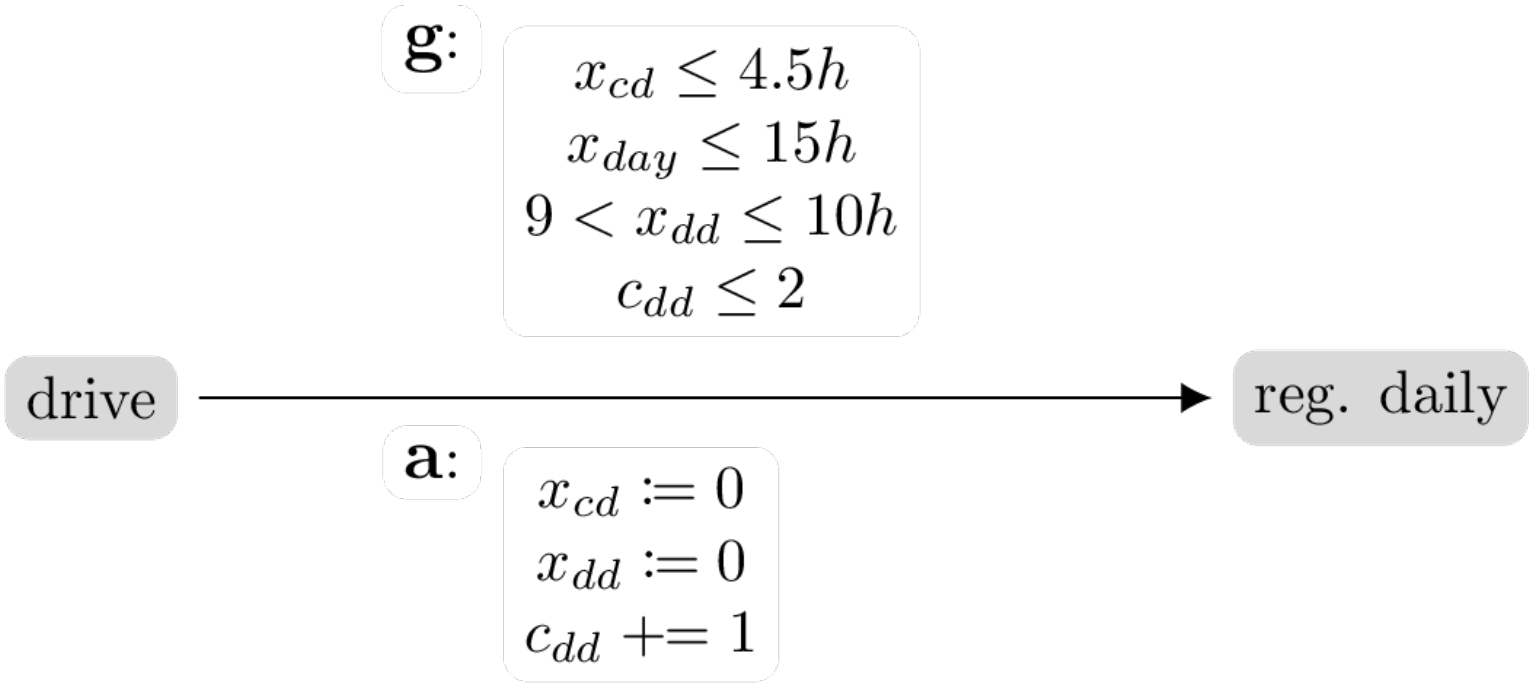
23 are nowhere active:

bits

counters

registers

$>$ 100 transitions



Expressivity and model-checking

Theorem

A set of words is accepted by an SWA iff it is definable in MSO.

Theorem

There is an algorithm that decides

Input: stopwatch automaton $\mathbb{A}$ and a word w over Σ

Problem: does $\mathbb{A}$ accepts w ?

in time

$$O\big(|\mathbb{A}|^2 \cdot t^x \cdot |w|\big)$$

where

$t :=$ *largest stopwatch bound of* $\mathbb{A}$

$x :=$ *number of stopwatches of* $\mathbb{A}$

Consistency-checking

Theorem

There is an algorithm that decides

Input: SWAs $\mathbb{A}, \mathbb{B}$

Problem: is there a word accepted by both $\mathbb{A}$ *and* $\mathbb{B}$ *?*

in time

$O(|\mathbb{A}|^3 \cdot |\mathbb{B}|^3 \cdot t^x \cdot s^y)$

where

$t :=$ *largest stopwatch bound of* $\mathbb{A}$

$x :=$ *number of stopwatches of* $\mathbb{A}$

$s :=$ *largest stopwatch bound of* $\mathbb{B}$

$y :=$ *number of stopwatches of* $\mathbb{B}$

Scheduling

Theorem

There is an algorithm that decides

Input: SWA $\mathbb{A}$*, letter* $a \in \Sigma$*, word* w *over* Σ*,* $n \in \mathbb{N}$

Problem: compute length n *word* v *over* Σ *such that*

$\mathbb{A}$ *accepts* wv

v *maximizes* $\#_a(v)$

in time

$$O\big(|\mathbb{A}|^2 \cdot t^x \cdot (|w| + n)\big)$$

where

$t :=$ *largest stopwatch bound of* $\mathbb{A}$

$x :=$ *number of stopwatches of* $\mathbb{A}$

Lower bound

Know: MC(Σ^*, SWA) decidable in time $O\big(|\mathbb{A}|^2 \cdot t^x \cdot |w|\big)$

Doubt: Is t^x tolerable? Can it be improved?

Interesting instances have large t and small x.

Question: replace t^x by $100^{100 \cdot x} \cdot t^{100}$?

Theorem

Assume FPT $\neq$ W[1]. Let $f : \mathbb{N} \to \mathbb{N}$ be a computable function.
Then MC(Σ^*, SWA) cannot be decided in time

$$\big(|\mathbb{A}| \cdot f(x) \cdot t \cdot |w|\big)^{O(1)}.$$

Question: Can we hardwire large constants in the data structure using Hybrid Modal Logic?

Some further development

- Study if data-representation can improve complexity of model checking;

Some further development

- Study if data-representation can improve complexity of model checking;
- Define/carve out a language that corresponds naturally to bounded stopwatch automata;

Some further development

- Study if data-representation can improve complexity of model checking;
- Define/carve out a language that corresponds naturally to bounded stopwatch automata;
- Formally verify the meta-theorems of model checking;

Some further development

- Study if data-representation can improve complexity of model checking;
- Define/carve out a language that corresponds naturally to bounded stopwatch automata;
- Formally verify the meta-theorems of model checking;
- Write a formally verified implementation of the model checker;

Some further development

- Study if data-representation can improve complexity of model checking;
- Define/carve out a language that corresponds naturally to bounded stopwatch automata;
- Formally verify the meta-theorems of model checking;
- Write a formally verified implementation of the model checker;
- Provide a front-end interface to build your law yourself;

Some further development

- Study if data-representation can improve complexity of model checking;
- Define/carve out a language that corresponds naturally to bounded stopwatch automata;
- Formally verify the meta-theorems of model checking;
- Write a formally verified implementation of the model checker;
- Provide a front-end interface to build your law yourself;
- Translate automata to semi-natural language description of the regulation;

Some further development

- Study if data-representation can improve complexity of model checking;
- Define/carve out a language that corresponds naturally to bounded stopwatch automata;
- Formally verify the meta-theorems of model checking;
- Write a formally verified implementation of the model checker;
- Provide a front-end interface to build your law yourself;
- Translate automata to semi-natural language description of the regulation;
- etc.

Some further development

- Study if data-representation can improve complexity of model checking;
- Define/carve out a language that corresponds naturally to bounded stopwatch automata;
- Formally verify the meta-theorems of model checking;
- Write a formally verified implementation of the model checker;
- Provide a front-end interface to build your law yourself;
- Translate automata to semi-natural language description of the regulation;
- etc.
- etc.

Thanks