

Controlled Natural Language as a Bridge between Large Language Models and Formal Methods and Programs

Joost J. Joosten

Universitat de Barcelona

Universitat Oberta de Catalunya, Barcelona

International Conference on Improving Administrative Decisions Using
AI, Experimentation and Sandboxes, Thursday, June 26



FV Time Manager
Formally Verified Time Library



DOCTORATS INDUSTRIALS
talent alliance!



Generalitat de Catalunya



Bosch i Gimpera
UNIVERSITAT DE BARCELONA



UNIVERSITAT DE BARCELONA



EUROPEAN COMMISSION
DIGITAL AND INDUSTRIAL TRANSFORMATION



EUROPEAN UNION



PROMETHEUS Group
INSTITUT D'INVESTIGACIÓ EN CIÈNCIES DE LA COMPUTACIÓ

Summary of this talk

- Legal practice shapes legal language

Summary of this talk

- Legal practice shapes legal language
- Automated or AI assisted decision making increasingly pertains to legal practice

Summary of this talk

- Legal practice shapes legal language
- Automated or AI assisted decision making increasingly pertains to legal practice
- Ergo: the corresponding languages (formal, query languages, etc.) impact legal language

Summary of this talk

- Legal practice shapes legal language
- Automated or AI assisted decision making increasingly pertains to legal practice
- Ergo: the corresponding languages (formal, query languages, etc.) impact legal language
- Natural language is one extreme of a spectrum of languages

Summary of this talk

- Legal practice shapes legal language
- Automated or AI assisted decision making increasingly pertains to legal practice
- Ergo: the corresponding languages (formal, query languages, etc.) impact legal language
- Natural language is one extreme of a spectrum of languages
- Formal language another

Summary of this talk

- Legal practice shapes legal language
- Automated or AI assisted decision making increasingly pertains to legal practice
- Ergo: the corresponding languages (formal, query languages, etc.) impact legal language
- Natural language is one extreme of a spectrum of languages
- Formal language another
- Controlled natural languages are in between

Summary of this talk

- Legal practice shapes legal language
- Automated or AI assisted decision making increasingly pertains to legal practice
- Ergo: the corresponding languages (formal, query languages, etc.) impact legal language
- Natural language is one extreme of a spectrum of languages
- Formal language another
- Controlled natural languages are in between
- Their usage has advantages and disadvantages

The Babylonian Confusion



- Pieter Bruegel de Oudere
(1525/1530 – 1569)

The Babylonian Confusion



- Pieter Bruegel de Oudere (1525/1530 – 1569)
- Stopping King Nemrod

The Babylonian Confusion



- Pieter Bruegel de Oudere (1525/1530 – 1569)
- Stopping King Nemrod
- To babble

The Babylonian Confusion



- Pieter Bruegel de Oudere (1525/1530 – 1569)
- Stopping King Nemrod
- To babble
- Fer a la babalà

From Arab: alà bāb Al-lāh, íd., literalment 'sobre la porta de Déu' or from Babel?

What is the Right Language for Law?



- Gottfried Leibniz (1646 – 1716)

What is the Right Language for Law?



- Gottfried Leibniz (1646 – 1716)
- Universal language

What is the Right Language for Law?



- Gottfried Leibniz (1646 – 1716)
- Universal language
- Characteristica Universali

What is the Right Language for Law?



- Gottfried Leibniz (1646 – 1716)
- Universal language
- Characteristica Universali
- Calculus Ratiocinator

What is the Right Language for Law?



- Gottfried Leibniz (1646 – 1716)
- Universal language
- Characteristica Universali
- Calculus Ratiocinator
- Utopian Dream

What is the Right Language for Law?



- Gottfried Leibniz (1646 – 1716)
- Universal language
- Characteristica Universali
- Calculus Ratiocinator
- Utopian Dream
- At best, domain specific languages

Practice and usage shapes language

- We create the words we need



Practice and usage shapes language

- We create the words we need
- Dutch has many words for different kinds of rain



Practice and usage shapes language

- We create the words we need
- Dutch has many words for different kinds of rain
- Famous fridgeon: any particle in the universe is a fridgeon exactly while Fodor's fridge is switched **on** (Janet Dean Fodor (1942 – 2023), Modules, Frames, Fridgeons, Sleeping Dogs, and the Music of the Spheres, 1987.).



Practice and usage shapes language



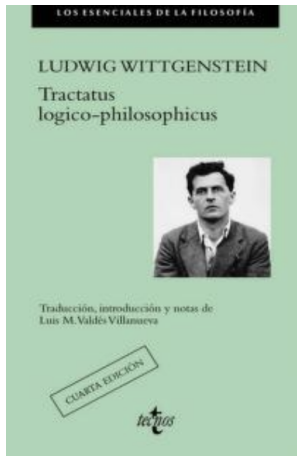
- We create the words we need
- Dutch has many words for different kinds of rain
- Famous fridgeon: any particle in the universe is a fridgeon exactly while Fodor's fridge is switched on (Janet Dean Fodor (1942 – 2023), Modules, Frames, Fridgeons, Sleeping Dogs, and the Music of the Spheres, 1987.).
- Cultural imprints: hefty language to impress

Practice and usage shapes language



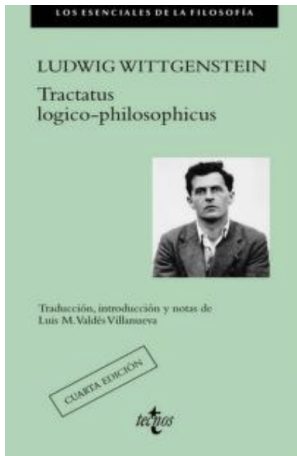
- We create the words we need
- Dutch has many words for different kinds of rain
- Famous fridgeon: any particle in the universe is a fridgeon exactly while Fodor's fridge is switched **on** (Janet Dean Fodor (1942 – 2023), Modules, Frames, Fridgeons, Sleeping Dogs, and the Music of the Spheres, 1987.).
- Cultural imprints: hefty language to impress
- Sophisticated :-)

Philosophy of Language: The Language Game



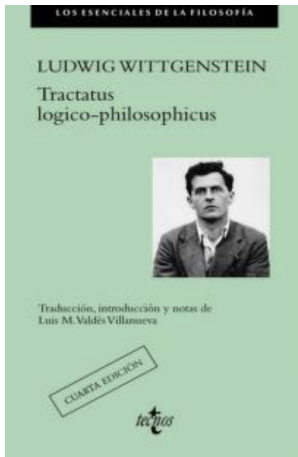
- Logical Positivism

Philosophy of Language: The Language Game



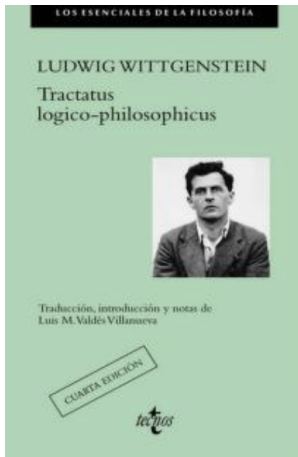
- Logical Positivism
- Verifiable relation between language and reality

Philosophy of Language: The Language Game



- Logical Positivism
- Verifiable relation between language and reality
- Wittgenstein II: Language as a game, no semantic link

Philosophy of Language: The Language Game



- Logical Positivism
- Verifiable relation between language and reality
- Wittgenstein II: Language as a game, no semantic link
- Quite similar to Current LLMs.

Law and Code



- Practice and usage shapes language

Law and Code



- Practice and usage shapes language
- Law often essentially needs discretionary powers when applied

Law and Code



- Practice and usage shapes language
- Law often essentially needs discretionary powers when applied
- Hence, open texture is needed

Law and Code



- Practice and usage shapes language
- Law often essentially needs discretionary powers when applied
- Hence, open texture is needed
- Any automated process and in particular, any automated process in the legal sector need unambiguity

Law and Code



- Practice and usage shapes language
- Law often essentially needs discretionary powers when applied
- Hence, open texture is needed
- Any automated process and in particular, any automated process in the legal sector need unambiguity
- The programmer needs to disambiguate?

An Example from a recent paper

- What can possibly be wrong with the following???:

arXiv > cs > arXiv:2503.09129

Search...
Help | Adv

Computer Science > Computers and Society

[Submitted on 12 Mar 2025]

Specification languages for computational laws versus basic legal principles

Petia Guintchev, Joost J. Joosten, Sofia Santiago Fernández, Eric Sancho Adamson, Aleix Solé Sánchez, Marta Soria Heredia

We speak of a \textit{computational law} when that law is intended to be enforced by software through an automated decision-making process. As digital technologies evolve to offer more solutions for public administrations, we see an ever-increasing number of computational laws. Traditionally, law is written in natural language. Computational laws, however, suffer various complications when written in natural language, such as underspecification and ambiguity which lead to a diversity of possible interpretations to be made by the coder

An Example from a recent paper

arXiv > cs > arXiv:2503.09129

Computer Science > Computers and Society

[Submitted on 12 Mar 2025]

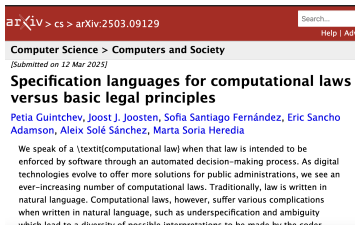
Specification languages for computational laws versus basic legal principles

Petia Guintchev, Joost J. Joosten, Sofia Santiago Fernández, Eric Sancho Adamson, Aleix Solé Sánchez, Marta Soria Heredia

We speak of a \textit{computational law} when that law is intended to be enforced by software through an automated decision-making process. As digital technologies evolve to offer more solutions for public administrations, we see an ever-increasing number of computational laws. Traditionally, law is written in natural language. Computational laws, however, suffer various complications when written in natural language, such as underspecification and ambiguity which lead to a diversity of possible interpretations to be made by the reader.

- What can possibly be wrong with the following???:
- Article 6.1.: The daily driving time shall not exceed nine hours. However, the daily driving time may be extended to at most 10 hours not more than twice during the week.

An Example from a recent paper



arXiv > cs > arXiv:2503.09129

Search... Help | Adv

Computer Science > Computers and Society

[Submitted on 12 Mar 2025]

Specification languages for computational laws versus basic legal principles

Petia Guintchev, Joost J. Joosten, Sofia Santiago Fernández, Eric Sancho Adamson, Aleix Solé Sánchez, Marta Soria Heredia

We speak of a \textit{computational law} when that law is intended to be enforced by software through an automated decision-making process. As digital technologies evolve to offer more solutions for public administrations, we see an ever-increasing number of computational laws. Traditionally, law is written in natural language. Computational laws, however, suffer various complications when written in natural language, such as underspecification and ambiguity which lead to a diversity of possible interpretations to be made by the reader.

- What can possibly be wrong with the following???:
- Article 6.1.: The daily driving time shall not exceed nine hours. However, the daily driving time may be extended to at most 10 hours not more than twice during the week.
- EU Regulation 561/2006 on road transport

Small leap in a year, giant step for a truck-driver

- Art. 4. (i) '**a week**' means the period of time between 00.00 on Monday and 24.00 on Sunday;



Small leap in a year, giant step for a truck-driver

- Art. 4. (i) *'a week' means the period of time between 00.00 on Monday and 24.00 on Sunday;*
- Art. 4. (k): *'daily driving time' means the total accumulated driving time between the end of one daily rest period and the beginning of the following daily rest period or between a daily rest period and a weekly rest period;.*



Small leap in a year, giant step for a truck-driver

- Art. 4. (i) '**a week**' means the period of time between 00.00 on Monday and 24.00 on Sunday;
- Art. 4. (k): '**daily driving time**' means the total accumulated driving time between the end of one daily rest period and the beginning of the following daily rest period or between a daily rest period and a weekly rest period;.
- Daily in daily drivingtime is not a substantive modifier, rather *daily drivingtime* is a privative phrase



Small leap in a year, giant step for a truck-driver



- Art. 4. (i) **'a week'** means the period of time between 00.00 on Monday and 24.00 on Sunday;
- Art. 4. (k): **'daily driving time'** means the total accumulated driving time between the end of one daily rest period and the beginning of the following daily rest period or between a daily rest period and a weekly rest period;.
- Daily in daily drivingtime is not a subsecutive modifier, rather *daily drivingtime* is a privative phrase
- Weekly in weekly rest is a subsecutive modifier, but of the informal notion rather than of the formal/technical one

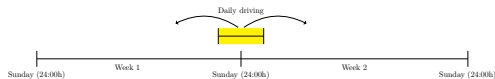
Small leap in a year, giant step for a truck-driver



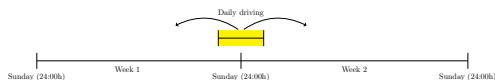
- Art. 4. (i) *'a week' means the period of time between 00.00 on Monday and 24.00 on Sunday;*
- Art. 4. (k): *'daily driving time' means the total accumulated driving time between the end of one daily rest period and the beginning of the following daily rest period or between a daily rest period and a weekly rest period;*
- *Daily* in *daily drivingtime* is not a subsecutive modifier, rather *daily drivingtime* is a privative phrase
- *Weekly* in *weekly rest* is a subsecutive modifier, but of the informal notion rather than of the formal/technical one
- UTC and 27 leap seconds

Small leap in a year, giant step for a truck-driver

- Underspecifaction

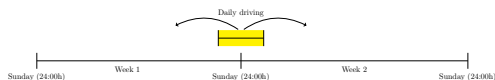


Small leap in a year, giant step for a truck-driver



- Underspecifaction
- Are we free to optimise?
Some software does.
With or without leap seconds?

Small leap in a year, giant step for a truck-driver

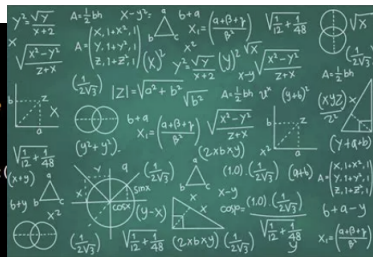


- Underspecifaction
- Are we free to optimise?
Some software does.
With or without leap seconds?
- Non-locality!

Languages without ambiguity

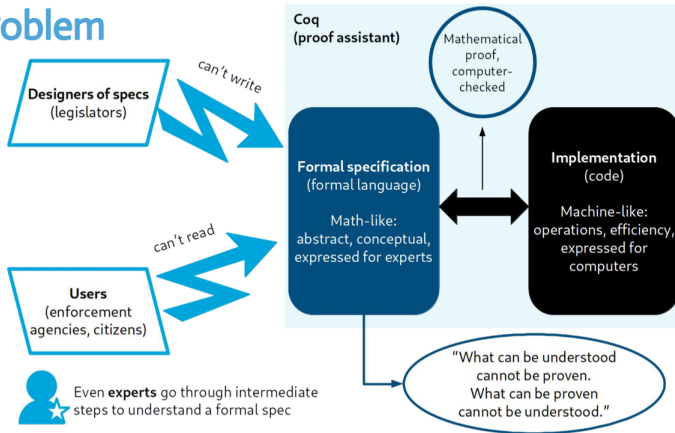
```
88 speakArticle = function(article){
89   wordArray = article.match(/\\S+\\s*/g);
90   phraseArray = [''];
91   var y = 0;
92   var i = 1;
93   length = wordArray.length //set once since we
94   while (i <= length) {
95     while (i % 40 != 0) {
96       if (wordArray.length > 0){
97         phraseArray[y] += wordArray.shift()
98         i++
99       }
100      else_break;
101    }
102    i++;
103    y++;

```



Legal Certainty, Explainability, Contestability

Problem



Through the Looking Glass: What is an explanation?



"There's no use trying," she said: "one can't believe impossible things." "I daresay you haven't had much practice," said the Queen. "When I was your age, I always did it for half-an-hour a day. Why, sometimes I've believed as many as six impossible things before breakfast."

Can code be the law?

TENSION TABLE:

Computable laws:

Language, software paradigm and legal principles

	Specification Language	Programming paradigm	Legal Principles		
			Legal Certainty	Accountability	Contestability
More accurate and exact but less understandable for the general public ↓	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

Conclusions Table		
	Natural Language	Formal Language
Legality	Subsubsection 4.2.1 - SALA may necessitate a prior legal authorization; - possible externalization of administrative ADM without pre-established legal procedures and guarantees; - more prone to generating erroneous software + fits traditional legal practice	Subsubsection 5.2.1 - current legal eco-system largely incompatible + software more likely to be according to specification + allows for formal methods certifying correctness
Legal Certainty	Subsubsection 4.2.2 - difference between <i>daily-life</i> and legal definitions of the terms - underspecification of the terms, leaving wide margins of interpretability + naturally embeds in citizen's reference frame	Subsubsection 5.2.2 - incomprehensibility of the formal language for the citizenry - explainability of the software does not necessarily entail the ability to accurately predict its behavior + no underspecification nor ambiguity
Arbitrariness	Subsubsection 4.2.3 - ambiguities and contradictions are resolved without further justifying the option taken as to the meaning of the ambiguous term	Subsubsection 5.2.3 + fully deterministic
Motivation	Subsubsection 4.2.4 + natural language can be used to provide a motivation; - motivation not necessarily ties up with the functioning of the automated implemented reasoning	Subsubsection 5.2.4 - execution of a software does not provide an explanation; + proof with formal methods of correctness can partly mitigate lack of motivation;
Clarity	Subsubsection 4.2.5 + easy to understand by the citizenry; -SALA may constitute a black box	Subsubsection 5.2.5 - unintelligibility of the regulation for most citizens - difficulties in retrieving the data driving the outcome

Table 1: Abbreviation: SALA stands for Software Acting as Legal Agent. If there is a plus sign upfront the observation is considered a positive aspect. Likewise, negative aspects are preceded by a minus sign.

① 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

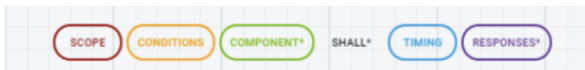
```
scope QualifiedEmployeeDiscount :  
  definition qualified_employee_discount  
    under condition is_property consequence equals  
      if employee_discount >$ customer_price *$ gross_profit_percentage then  
        customer_price *$ gross_profit_percentage  
      else employee_discount
```

Example FRET: Formal Requirement Elicitation Tool

An attempt at bridging formal and natural language

The screenshot shows the FRET tool interface. At the top, there is a toolbar with six buttons: SCOPE (red), CONDITIONS (orange), COMPONENT* (green), SHALL* (black), TIMING (blue), and RESPONSES* (purple). A question mark icon is on the right. Below the toolbar is a text input area containing the sentence: "in roll_hold mode RollAutopilot shall always satisfy autopilot_engaged & no_other_lateral_mode". The words in the sentence are color-coded to match the buttons: "in" (red), "roll_hold" (orange), "mode" (green), "RollAutopilot" (green), "shall" (black), "always" (blue), "satisfy" (black), "autopilot_engaged" (black), "&" (black), and "no_other_lateral_mode" (purple).

FRET under the logic loupe: TIMING

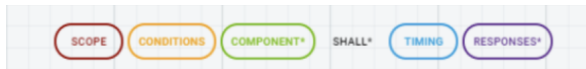


TIMING is optional and specifies when response is expected

- immediately
- never
- eventually (the default reading when Timing is omitted)
- always
- within n time units
- for n time units
- after n time units
- ...

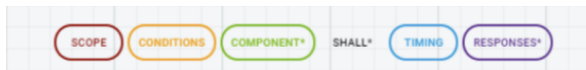
10 timing options

FRET under the logic loupe: SHALL



SHALL is mandatory

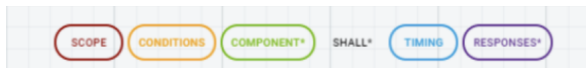
FRET under the logic loupe: COMPONENT



COMPONENT is mandatory

For example: The_Car

FRET under the logic loupe: CONDITION

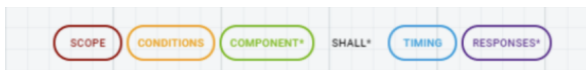


CONDITION is an optional feature: a Boolean expression

For example: When Traffic_Light_Is_Red & No_Police_Car_Nearby

Three options: void condition, trigger condition, continual

FRET under the logic loupe: SCOPE



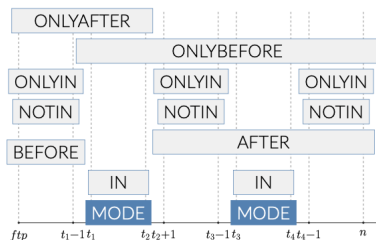
SCOPE is an optional feature: a finite collection of disjoint time intervals where the requirement is imposed

Built from a Boolean Mode M with an operator applied to it: $\mathcal{O}(M)$

- before M
- after M
- in M
- not in M
- only before M
- only after M
- only in M
- global (default scope when omitted)

8 operators in total

FRET under the logic loupe: SCOPE



FRET under the logic loupe: summary

Scope

Conditions

Component*

Shall*

Timing

Responses*

Mandatory Fields

Component system involved
Shall obligation keyword

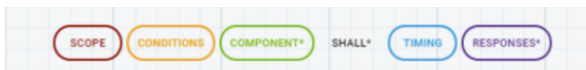
Response expected boolean behavior

Optional Fields

Timing when *response* should occur
Conditions applicability context: *void* (unconditional), *trigger* (activates on event), *continual* (holds while true)

Scope temporal intervals of enforcement built from a *mode* (boolean) and an operator

FRET under the logic loupe: Semantic templates



8 scope operators

Condition or no condition

7 Timing options

give rise to

$$8 \times 3 \times 10 = 240$$

so-called semantic templates.

Each of the form

$$G(\mathcal{O}(M) \wedge \mathcal{C} \rightarrow \mathcal{T}(R))$$

This is a simplified representation of the fragment of LTL covered by this version of FRET*

* (there are some small letters)

FRET interface: various language levels

Update Requirement

Requirement ID	Parent Requirement ID	Project
AP-002A		LM_requirements

Rationale and Comments

Rationale

Roll hold mode shall only be active, when the autopilot is engaged. Simultaneous engagement of several lateral modes shall not be possible.

Comments

Requirement Description

A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "**". For information on a field format, click on its corresponding bubble.

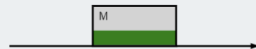


In roll_hold mode RollAutopilot shall always satisfy autopilot_engaged & no_other_lateral_mode

SEMANTICS

Semantics

ENFORCED: In every interval where **roll_hold** holds. TRIGGER: first point in the interval. REQUIRES: for every trigger, RES must hold at all time points between (and including) the trigger and the end of the interval.



$M = \text{roll_hold}$, Response = (autopilot_engaged & no_other_lateral_mode).

Diagram Semantics

Formalizations

Future Time LTL

$(\text{LAST } V \text{ (roll_hold} \rightarrow \text{(autopilot_engaged \& no_other_lateral_mode))})$

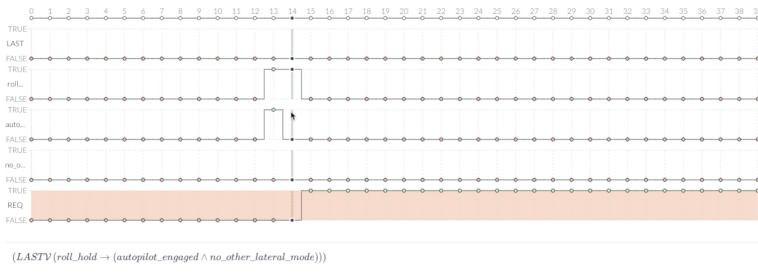
Target: **RollAutopilot** component.

Past Time LTL

$(H \text{ (roll_hold} \rightarrow \text{(autopilot_engaged \& no_other_lateral_mode))})$

Target: **RollAutopilot** component.

FRET: interactive sample testing



Users can test the behaviour of their FRET phrases

Simplified architecture

Ex: MTL Formulas Translating FRETISH for Future Time on Infinite Traces

$$\mathcal{G} \left(\llbracket \text{Mode} \rrbracket \wedge \text{Cond} \rightarrow \mathcal{T}_{\infty}(\text{Resp}, \llbracket \text{Mode} \rrbracket) \right) \quad \text{if Cond is continual}$$

$$\mathcal{G} \left(\text{ChangeTo} \left(\llbracket \text{Mode} \rrbracket \wedge \text{Cond} \right) \rightarrow \mathcal{T}_{\infty}(\text{Resp}, \llbracket \text{Mode} \rrbracket) \right) \quad \text{if Cond is trigger}$$

ChangeTo($\cdot$) Activates when expression changes from false to true to capture the trigger dynamics

$\llbracket \text{Mode} \rrbracket$ MTL translation of the Scope

$\mathcal{T}_{\infty}$ MTL translation of the Timing constraint

Simplified architecture: lookup tables

Ordinal	Scope operator	LTL definition
0	Global Mode	$\top$
1	In Mode	Mode
2	Not In Mode	$\neg \text{Mode}$
3	Only In Mode	$\neg \text{Mode}$
4	Before Mode	$\mathcal{H} \neg \text{Mode}$
5	Only Before Mode	$\mathcal{O} \text{ Mode}$
6	After Mode	$\mathcal{O} (\neg \text{Mode} \wedge \mathcal{Y} \text{ Mode})$
7	Only After Mode	$\mathcal{H} (\mathcal{Y} \text{ Mode} \rightarrow \text{Mode})$

(a) Look-up table for FRETISH Scope

ID	Condition kind
0	void
1	continual
2	trigger

(b) Look-up table for FRETISH Condition

ID	FRETISH Timing	MTL formula
0	eventually	$\llbracket \text{Mode} \rrbracket \mathcal{U} (\text{Resp} \wedge \llbracket \text{Mode} \rrbracket)$
1	immediately	Resp
2	next	$\mathcal{X} \text{ Resp} \vee \mathcal{X} \text{ ChangeTo} \neg \llbracket \text{Mode} \rrbracket$
3	always	$(\mathcal{X} \text{ ChangeTo} \neg \llbracket \text{Mode} \rrbracket) \mathcal{R} \text{ Resp}$
4	never	$(\mathcal{X} \text{ ChangeTo} \neg \llbracket \text{Mode} \rrbracket) \mathcal{R} \neg \text{Resp}$
5	within (d)	$\Diamond_{[0,d]} \text{Resp} \vee (\llbracket \text{Mode} \rrbracket \mathcal{U}_{[0,d]} \neg \llbracket \text{Mode} \rrbracket)$
6	for (d)	$\Box_{[0,d]} \text{Resp} \vee ((\text{Resp} \wedge \llbracket \text{Mode} \rrbracket) \mathcal{U}_{[0,d]} \neg \llbracket \text{Mode} \rrbracket)$
7	after(d)	$\left(\Box_{[0,d]} \neg \text{Resp} \wedge \Diamond_{[d+1,d+1]} \text{Resp} \right) \vee ((\neg \text{Resp} \wedge \llbracket \text{Mode} \rrbracket) \mathcal{U}_{[0,d+1]} \neg \llbracket \text{Mode} \rrbracket)$
8	until (stopCond)	$\Box \text{Resp} \vee \left(\text{Resp} \mathcal{U} (\text{stopCond} \vee \text{StrictChangeTo} (\neg \llbracket \text{Mode} \rrbracket)) \right)$
9	before (stopCond)	$(\text{Resp} \mathcal{R} \neg \text{stopCond}) \vee \left((\llbracket \text{Mode} \rrbracket \wedge \mathcal{X} \neg \llbracket \text{Mode} \rrbracket) \mathcal{R} \neg \text{stopCond} \right)$

(c) Look-up table for FRETISH Timing for infinite trace when the Scope does not include "only".

Comparison and Occam

FRETISH: In Scope upon Condition Component shall before StopCondition satisfy Response

FRET's MTL: $((G ((! (Scope) \& (X \text{ Scope}))) \mid (X (((Scope \& (X (! Scope))) \vee (((! Condition) \& ((X Condition) \& (! (Scope \& (X (! Scope)))))) \rightarrow ((X (! (((! (StopCondition) \& (Response \mid (Scope \& (X (! Scope)))))) \& (! (Scope \& (X (! Scope)))))) \vee StopCondition))) \& (! (Scope \& (X (! Scope)))))) \& (Condition \rightarrow (! (((! (StopCondition) \& (Response \mid (Scope \& (X (! Scope)))))) \& (! (Scope \& (X (! Scope)))))) \vee StopCondition)))))) \& (Scope \rightarrow (((Scope \& (X (! Scope))) \vee (((! Condition) \& ((X Condition) \& (! (Scope \& (X (! Scope)))))) \rightarrow ((X (! (((! (StopCondition) \& (Response \mid (Scope \& (X (! Scope)))))) \& (! (Scope \& (X (! Scope)))))) \vee StopCondition))) \& (! (Scope \& (X (! Scope)))))) \& (Condition \rightarrow (! (((! (StopCondition) \& (Response \mid (Scope \& (X (! Scope)))))) \& (! (Scope \& (X (! Scope)))))) \vee StopCondition))))))$

FRET translation

FRETISH: In Scope upon Condition Component shall before StopCondition satisfy Response

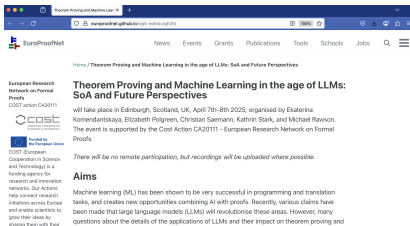
Simplified translation: $G (\text{ChangeTo} (Scope \& Condition) \rightarrow ((Response \vee ! StopCondition) \mid ((Scope \& X ! Scope) \vee ! StopCondition)))$

Our simplified translation (preparing Rocq implementation)

Spurious findings: the best/worst of both worlds?

- **Counter-intuitive constructs** whose informal interpretation diverges from their actual formal semantics, often leading to misunderstandings:
 - **TheParcel** shall **within 1 day** **satisfy BeDelivered** $\Rightarrow$ **TheParcel** shall **eventually satisfy BeDelivered**
 - **'Only In' Scopes defy logic**: Their semantics aren't logically grounded, but come from engineering intuition.
 - **TheDriver** shall **after 3 hours of driving** **satisfy Rest** forces that the rest cannot occur before 3 hours of driving.

Non-sharing incentives



16th century protagonists: Gerolamo Cardano, Niccolò Fontana (Tartaglia (stammerer)), Scipione del Ferro, Ludovico Ferrari, etc.

21st century protagonist: ChatGTP, DeepSeek, Mistral, Claude, ...

Belief revision appreciation revision

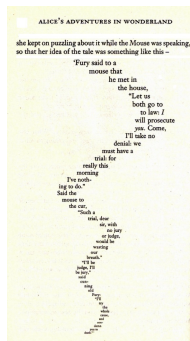


Success from information theory: Self-attention with triple (queries, keys, values) in encoders and decoders

Controlled Natural Languages are promising to bridge reasoning and LLMs.

FRET simplification apology: The simpler they are, the better, the closer the formalisation to the CNL the better

Applied Logic in Law



It keeps the logician off the street

Thanks

