Cyber-Physical Systems

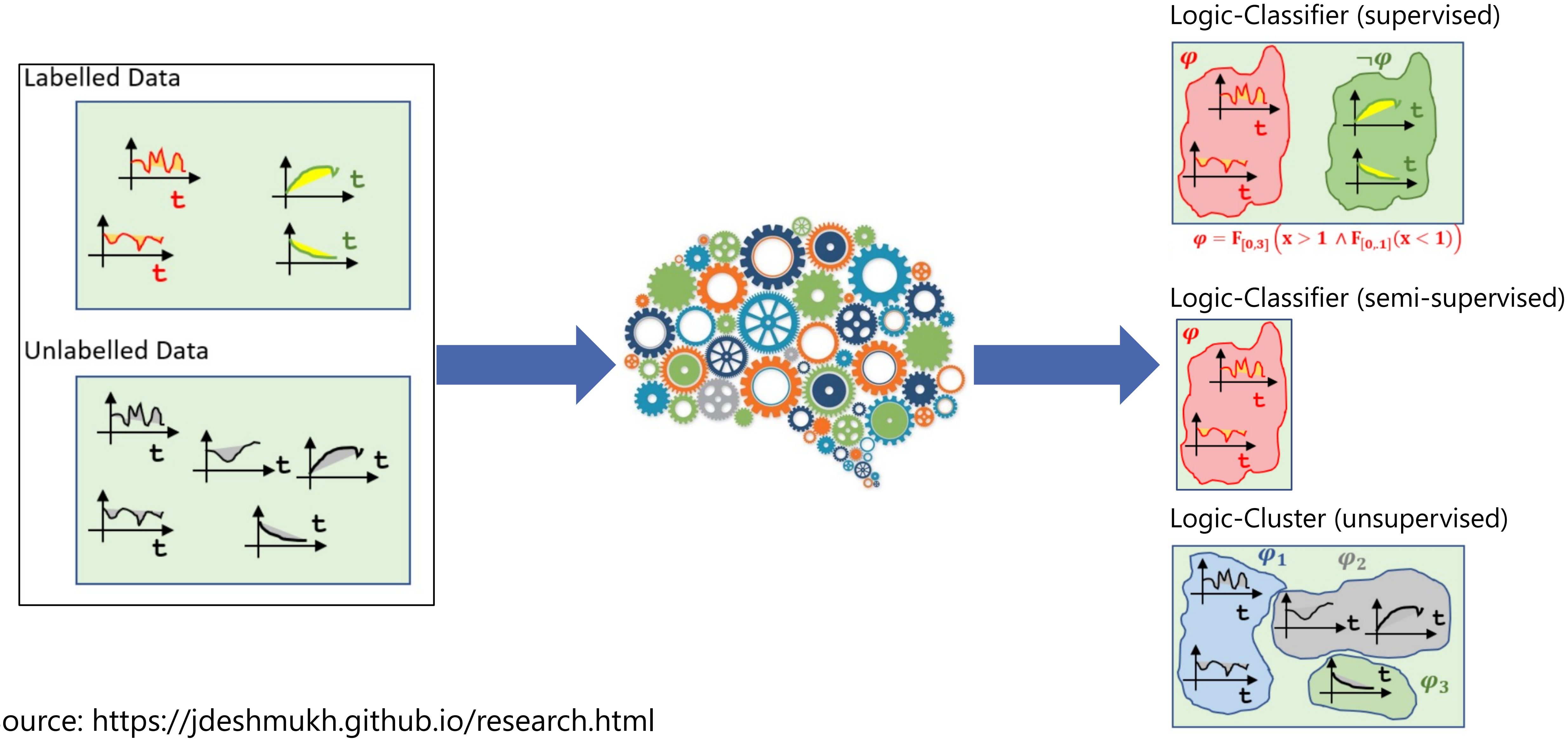
Laura Nenzi

Università degli Studi di Trieste

I Semestre 2023

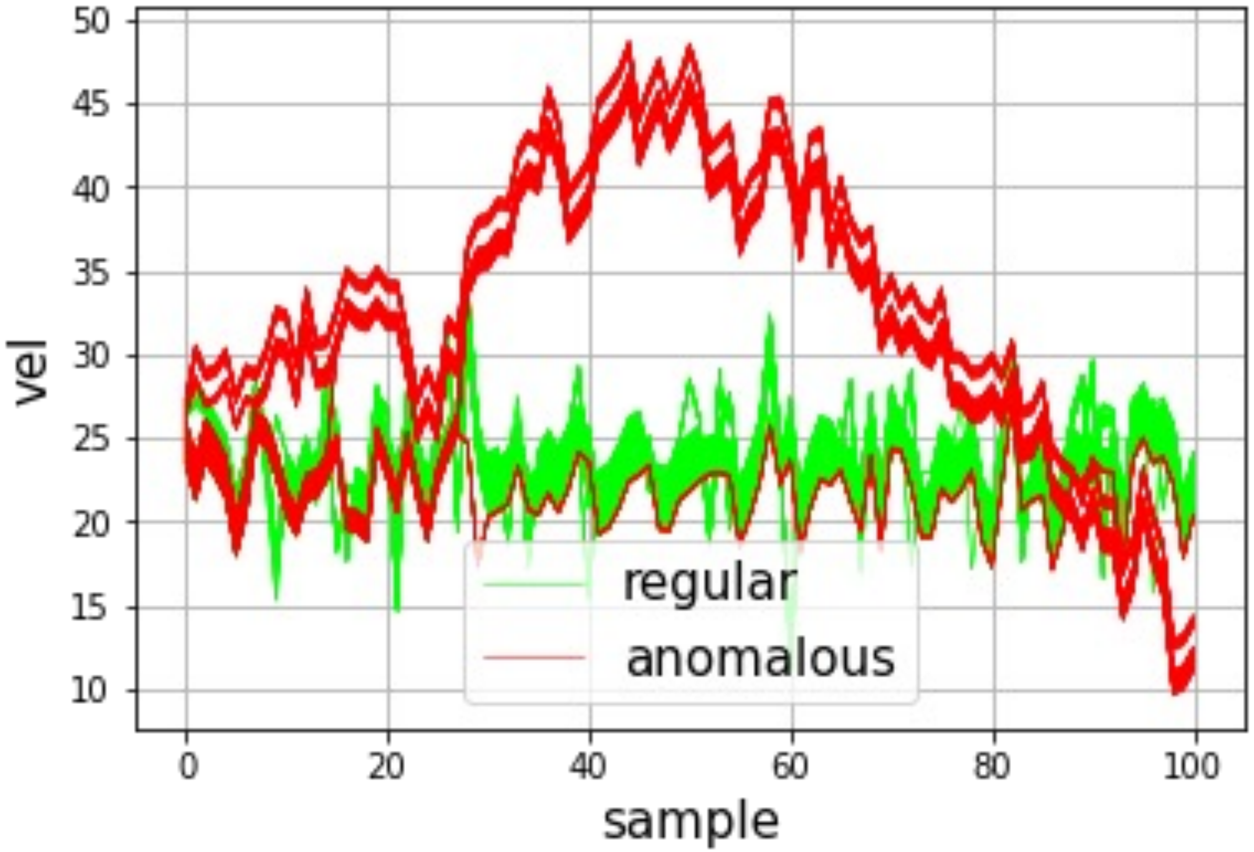
Lecture 24: TL Learning insights

Temporal Logic requirement mining



Source: https://jdeshmukh.github.io/research.html

STL Classifiers ((Semi-)Supervised Learning)



Goal: learning a specification/ classifier as a temporal logic formula to discriminate as much as possible between regular and anomalous behaviours.

We want to learn both the structure and the parameters of the formula

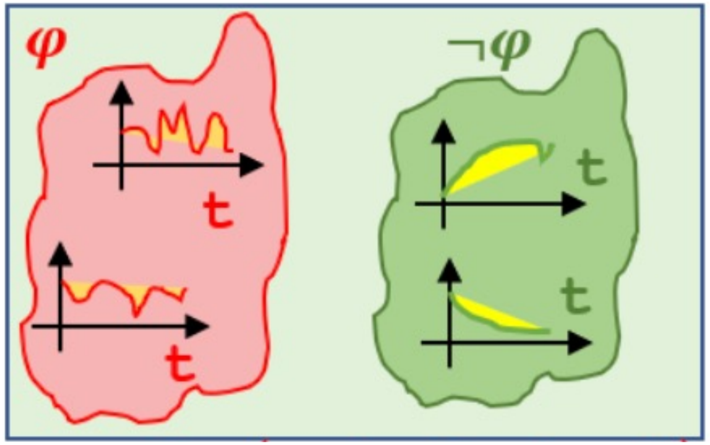
STL Classifier: Problem Statement

We want a way to search in the space of STL formulae considering training data X_{learn}

Supervised two-class classification problem

Training data set: two sets

- regular X^{+}_{learn}
- anomalous X^{-}_{learn}

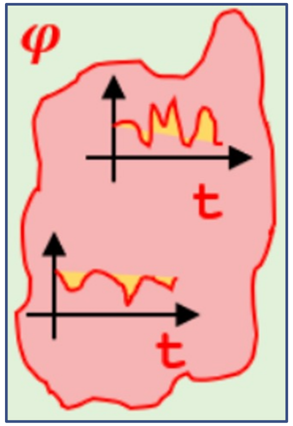


Find the best φ that better separates the two sets.

Semi-supervised one-class classification prob

Training data set: one set

- regular X^{+}_{learn}



Find the "tight" φ that is satisfied by the set

STL classifier (supervised): ROGE

- Bi-level algorithm:
  - learning formula structure via Genetic Programming (GP)
  - learn parameters of the formula using by Bayesian Optimisation

- A **fitness function** f measures the quality of candidate solutions and depends on the kind of problem at hand (two-classes, one-class)

$$f(\varphi; X^{+}_{\text{learn}}, X^{-}_{\text{learn}}) = -\frac{\mathbb{E}_{X^{+}_{\text{learn}}}(\rho_\varphi) - \mathbb{E}_{X^{-}_{\text{learn}}}(\rho_\varphi)}{\sigma_{\varphi, X^{+}_{\text{learn}}} + \sigma_{\varphi, X^{-}_{\text{learn}}}}$$

Require: $\mathcal{D}_p, \mathcal{D}_n, \mathbb{K}, Ne, Ng, \alpha, s$

$gen \leftarrow \text{GENERATEINITIALFORMULAE}(Ne, s)$

$gen_\Theta \leftarrow \text{LEARNINGPARAMETERS}(gen, G, \mathbb{K})$

for $i = 1 \dots Ng$ **do**

$\quad subg_\Theta \leftarrow \text{SAMPLE}(gen_\Theta, F)$

$\quad newg \leftarrow \text{EVOLVE}(subg_\Theta, \alpha)$

$\quad newg_\Theta \leftarrow \text{LEARNINGPARAMETERS}(newg, G, \mathbb{K})$

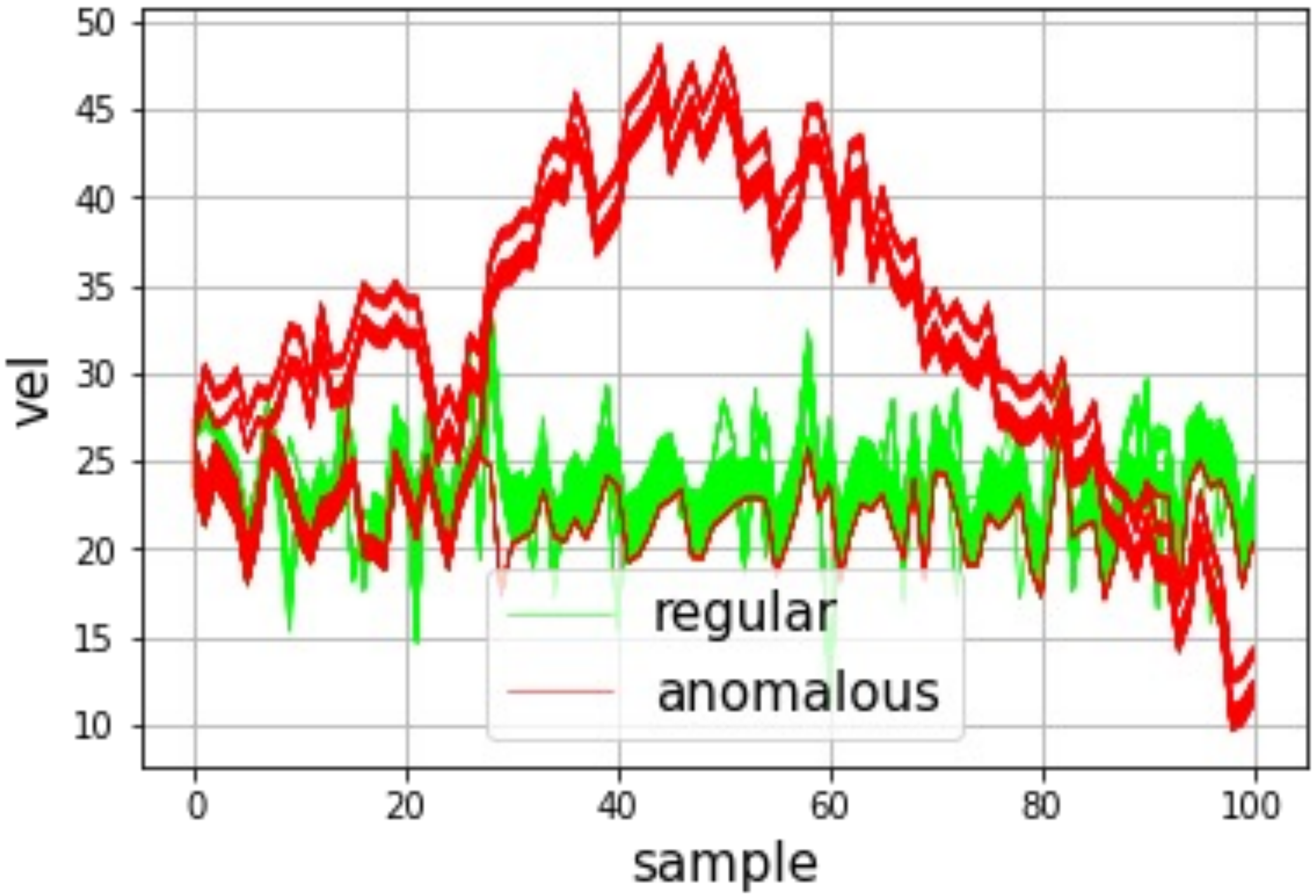
$\quad gen_\Theta \leftarrow \text{SAMPLE}(newg_\Theta \cup gen_\Theta, F)$

end for

return gen_Θ

[L. Nenzi, S. Silvetti, E. Bartocci, L. Bortolussi: A Robust Genetic Algorithm for Learning Temporal Specifications from Data. QEST 2018]

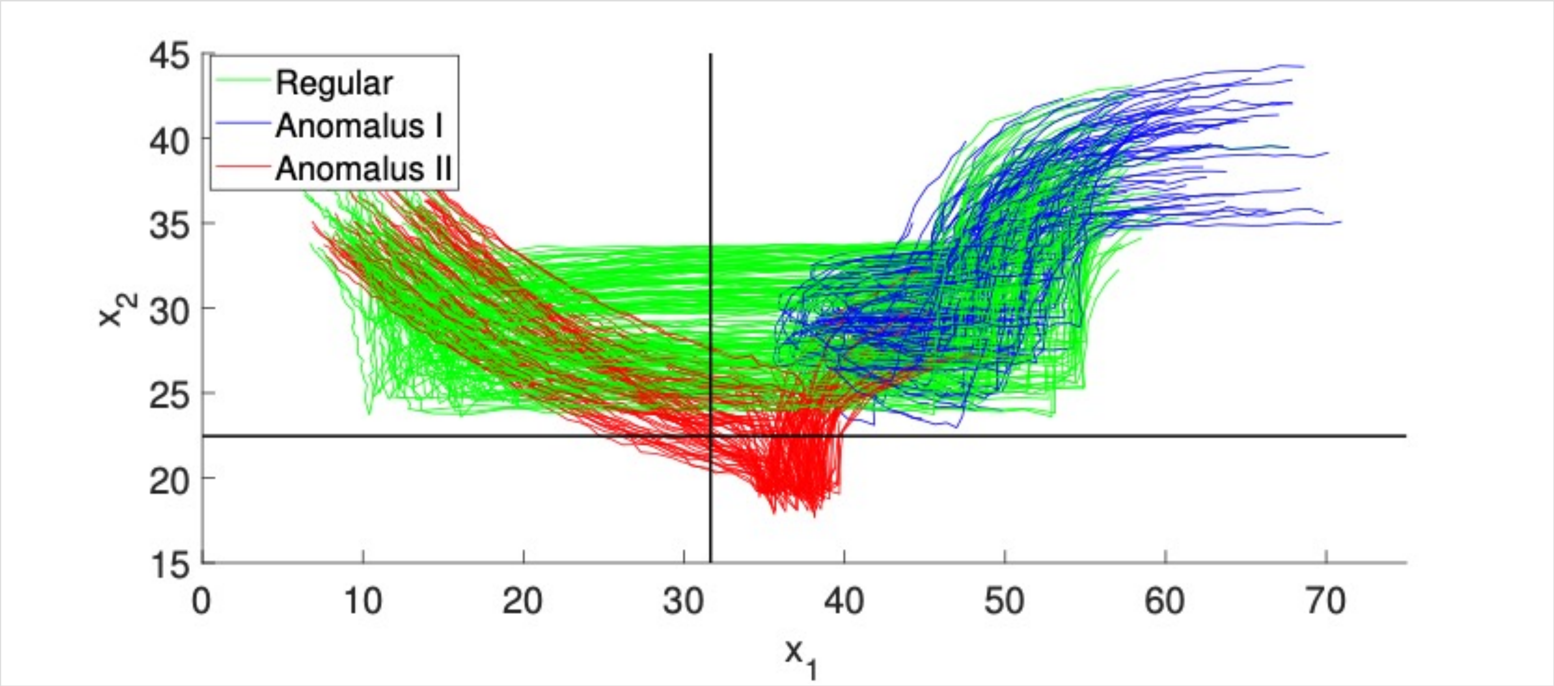
Results: Train Cruise



$$(F_{[22,40]}(\mathtt{vel} > 24.48)) \wedge (F_{[46,49]}(19.00 < \mathtt{vel} < 26.44))$$

Results: Maritime Surveillance

Synthetic dataset of naval surveillance of 2-dimensional coordinates traces of vessels behaviours.



$$((x_2 > 22.46)\,\mathcal{U}_{[49,287]}\,(x_1 \leq 31.65))$$

Limitation of ROGE

- Initial population designed ”by hand”
- The learning parameter algorithm can be slow (depending on the size parameter space)

STL Classifier: Context Free Grammar

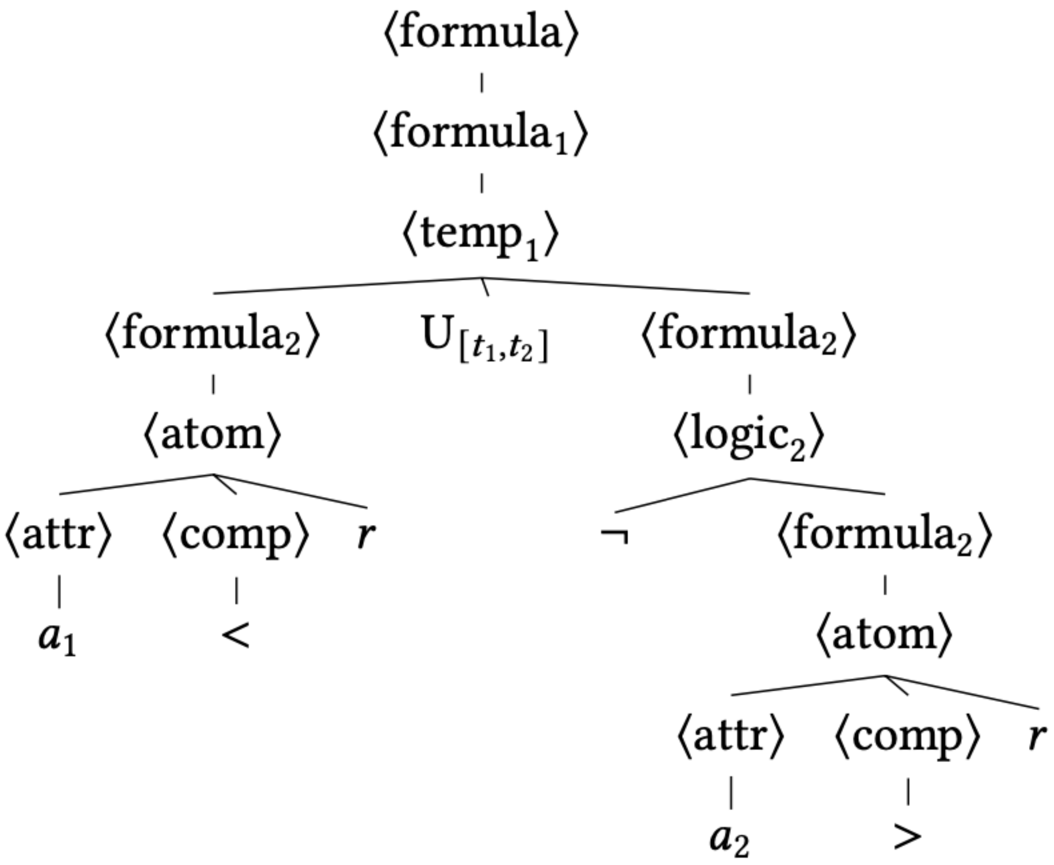
$$
\begin{aligned}
\langle \text{formula} \rangle &::= \langle \text{formula}_1 \rangle \\
\langle \text{formula}_i \rangle &::= \begin{cases} \langle \text{atom} \rangle \mid \langle \text{logic}_i \rangle \mid \langle \text{temp}_i \rangle & \text{if } i < i_{\text{max}} \\ \langle \text{atom} \rangle \mid \langle \text{logic}_i \rangle & \text{otherwise} \end{cases} \\
\langle \text{logic}_i \rangle &::= \neg \langle \text{formula}_i \rangle \mid \langle \text{formula}_i \rangle \wedge \langle \text{formula}_i \rangle \\
\langle \text{temp}_i \rangle &::= \langle \text{formula}_{i+1} \rangle \mathrm{U}_{\langle \text{interval} \rangle} \langle \text{formula}_{i+1} \rangle \mid \\
&\quad\ \mathrm{G}_{\langle \text{interval} \rangle} \langle \text{formula}_{i+1} \rangle \mid \\
&\quad\ \mathrm{F}_{\langle \text{interval} \rangle} \langle \text{formula}_{i+1} \rangle \\
\langle \text{interval} \rangle &::= [\langle \text{num} \rangle, \langle \text{num} \rangle] \\
\langle \text{atom} \rangle &::= \langle \text{attr} \rangle \langle \text{comp} \rangle 0.\langle \text{num} \rangle \\
\langle \text{attr} \rangle &::= a_1 \mid a_2 \mid \dots \mid a_{|A|} \\
\langle \text{comp} \rangle &::= \ < \ \mid \ > \\
\langle \text{num} \rangle &::= \langle \text{digit} \rangle \langle \text{digit} \rangle \\
\langle \text{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

[F. Pigozzi, E. Medvet, L. Nenzi. Mining Road Traffic Rules with Signal Temporal Logic and Grammar-Based Genetic Programming, Applied Sciences, 2022]

[F. Pigozzi, L. Nenzi., E. Medvet, BUSTLE: a Versatile Tool for the Evolutionary Learning of STL Specifications from Data (second revision on Evolutionary Computation]

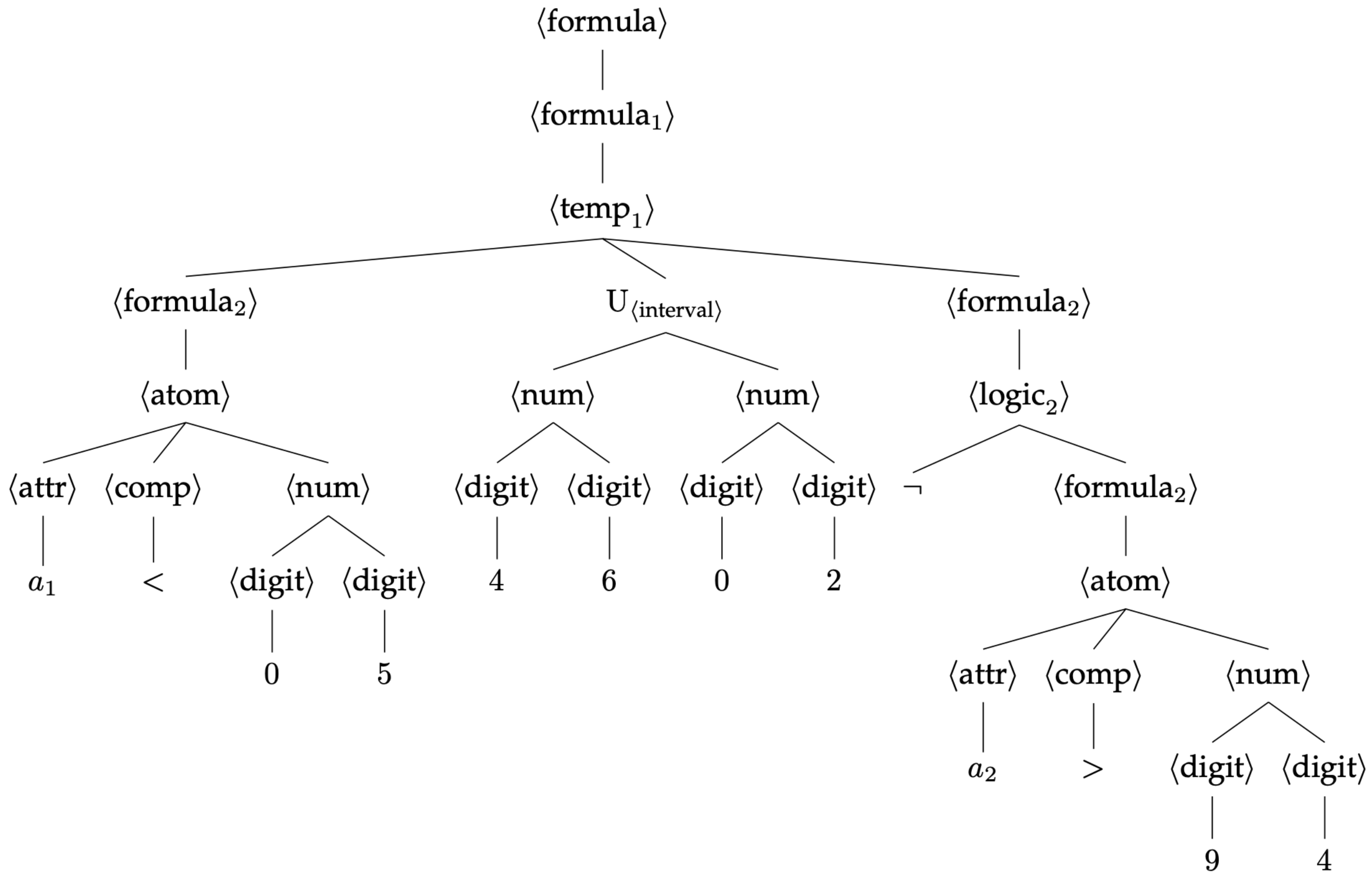
STL classifier: Building the population

- Candidate formulas are represented as derivation trees of a grammar

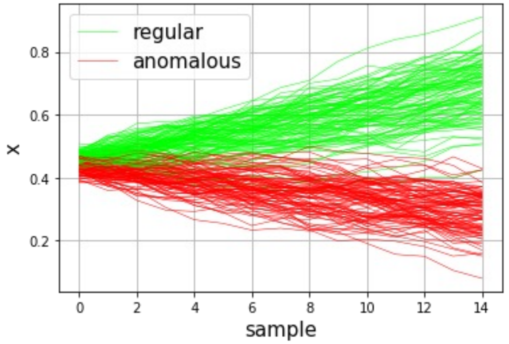


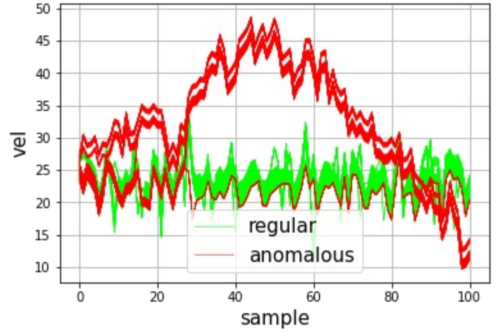
STL classifier: Building the population

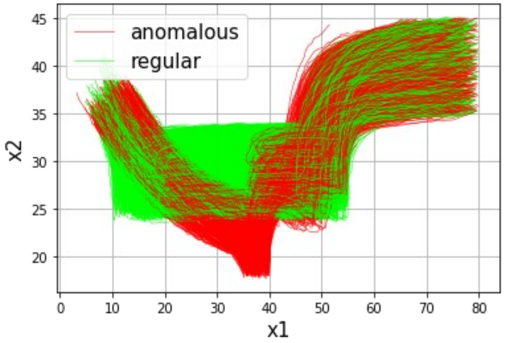
- Candidate formulas are represented as derivation trees of a grammar



Results

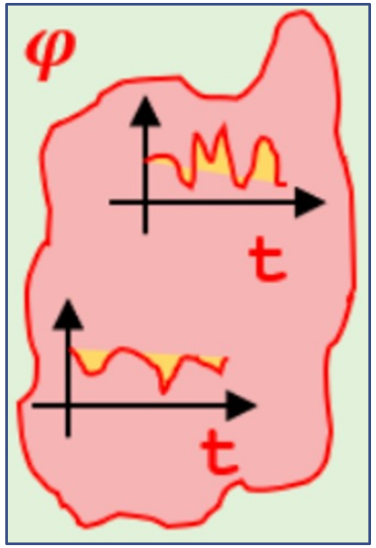






Dataset	Algorithm	FNR	FPR	Acc	Time
Linear	Random	0.20	0.20	0.80	11
	BUSTLE (single-level)	0.00	0.00	1.00	15
	BUSTLE (bi-level)	0.00	0.00	1.00	112
	Nenzi et al. (2018)	0.00	0.00	1.00	113
	Mohammadinejad et al. (2020b)	N/A	N/A	0.98	39
Train	Random	0.55	0.53	0.46	31
	BUSTLE (single-level)	0.03	0.05	0.96	26
	BUSTLE (bi-level)	0.00	0.03	0.98	523
	Nenzi et al. (2018)	0.10	0.00	0.95	576
	Mohammadinejad et al. (2020b)	N/A	N/A	0.98	32
Maritime	Random	0.52	0.50	0.49	84
	BUSTLE (single-level)	0.00	0.00	1.00	109
	BUSTLE (bi-level)	0.00	0.00	1.00	1477
	Nenzi et al. (2018)	0.00	0.00	1.00	1599
	Mohammadinejad et al. (2020b)	0.05	0.02	0.96	73
	Bombara and Belta (2021)	N/A	N/A	0.98	140

STL Classifier: Fitness Function for the one-class problem



Training data set: one set

- regular X^{+}_{learn}

Fitness, two high level requirements:

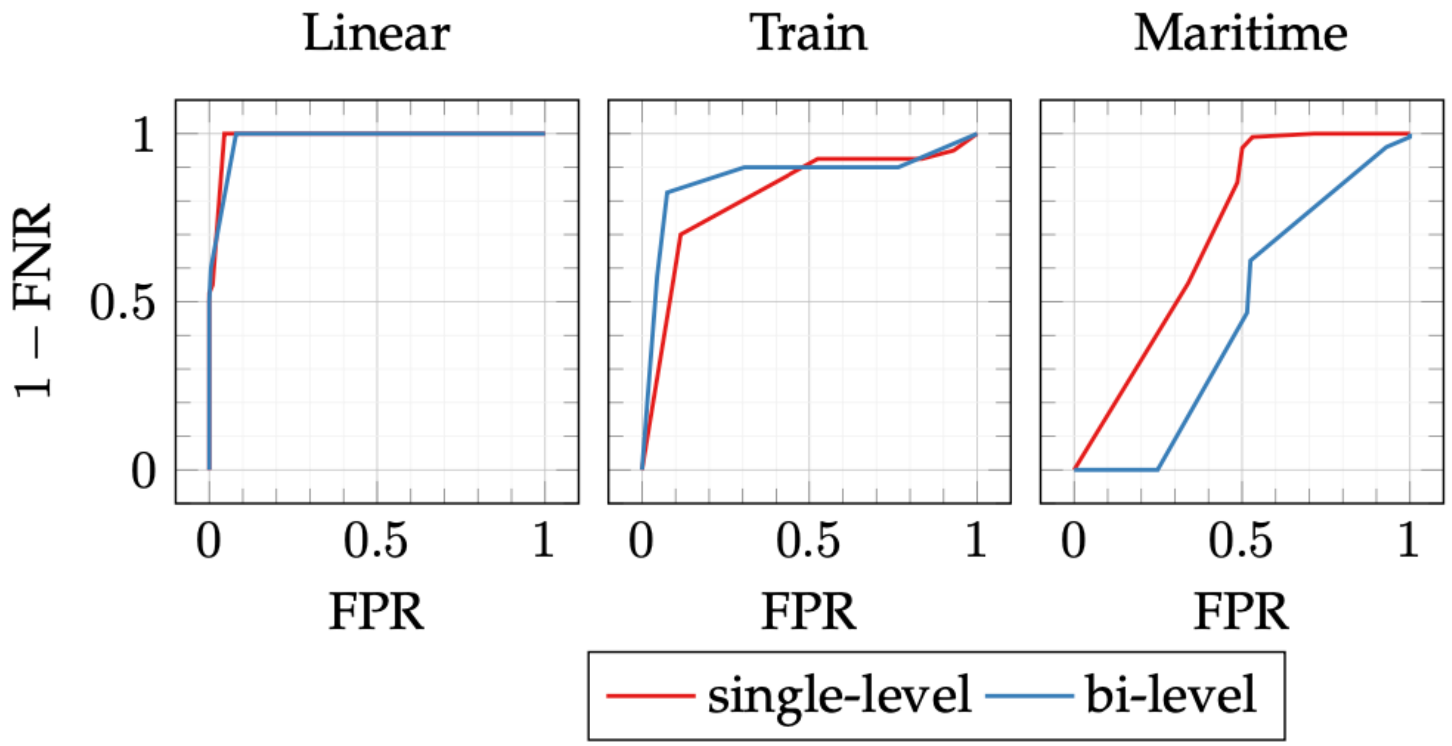
1. **Tight** formulas should be preferred
2. Formulas that lead to few false anomalies should be preferred

$$f(\varphi; X^{+}_{\text{learn}}) = \alpha \underbrace{\frac{1}{|X^{+}_{\text{learn}}|} \left|\{\boldsymbol{x} \in X^{+}_{\text{learn}} : \boldsymbol{x} \not\models \varphi\}\right|}_{2} + \overbrace{\frac{1}{\sigma'_{\varphi, X^{+}_{\text{learn}}} |X^{+}_{\text{learn}}|} \sum_{\boldsymbol{x} \in X^{+}_{\text{learn}}} |\rho(\varphi, \boldsymbol{x})|}^{1}$$

Results

	Variant	Two-classes					One-class				
		FNR	FPR	Acc	Time	c	FNR	FPR	Acc	Time	c
Lin.	Random	0.20	0.20	0.80	11	8.0	0.98	0.20	0.41	10	8.0
	BUSTLE (single-l.)	0.00	0.00	1.00	15	9.5	0.45	0.00	0.77	11	11.0
	BUSTLE (bi-l.)	0.00	0.00	1.00	112	12.5	0.40	0.00	0.80	145	11.0
Train	Random	0.55	0.53	0.46	31	8.0	0.81	0.15	0.52	18	8.0
	BUSTLE (single-l.)	0.03	0.05	0.96	26	12.0	0.30	0.12	0.79	25	11.0
	BUSTLE (bi-l.)	0.00	0.03	0.98	523	13.0	0.18	0.08	0.87	438	13.5
Marit.	Random	0.52	0.50	0.49	84	8.0	0.77	0.21	0.51	73	8.0
	BUSTLE (single-l.)	0.00	0.00	1.00	109	9.5	0.15	0.49	0.68	72	9.5
	BUSTLE (bi-l.)	0.00	0.00	1.00	1477	9.0	0.38	0.52	0.55	2008	12.0

Results



Limitations:

- There may be several good classifiers
- Finding the best classifier might be unfeasible
- There may not exist a single, good classifier

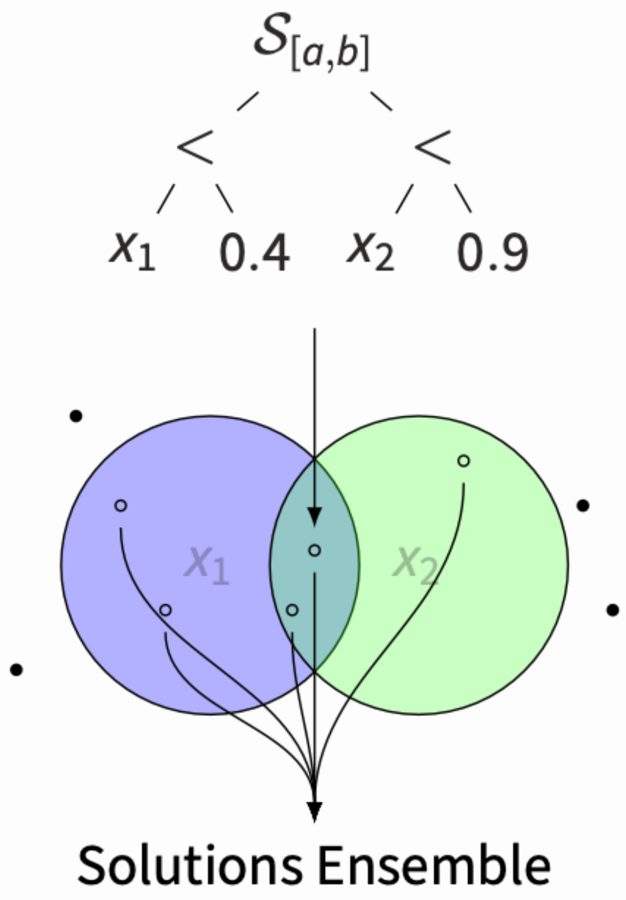
A one-shot algorithm

An evolutionary algorithm that learns an ensemble of solutions in a single run

- Population update:
  - Divide population in groups, one for each variable
  - The fittest formula of each group goes to next generation (elitism)
  - The remaining offspring is obtained reproducing the individuals

- Solutions update. If some individuals solve the problem (f $< \epsilon$), consider their groups:
  - Remove from the population the individuals in these groups (extinction)
  - Add them to the solutions ensemble
  - Refill the population with new individuals (random immigrants)

Stop once n_{target} variables have been solved



[Patrick Indri, Alberto Bartoli, Eric Medvet, Laura Nenzi: One-Shot Learning of Ensembles of Temporal Logic Formulas for Anomaly Detection in Cyber-Physical Systems. EuroGP 2022: 34-50]

An online application

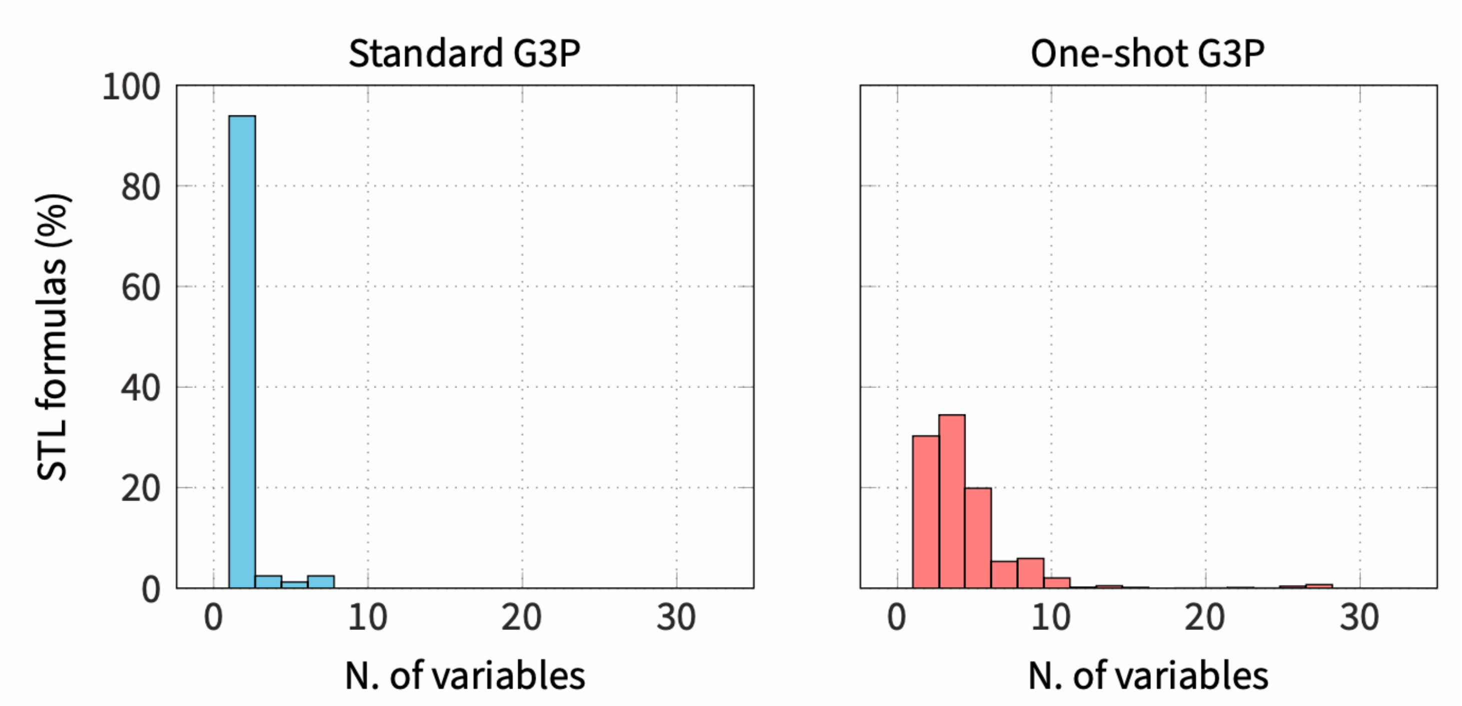
- For “online” anomaly detection
- using Past STL
- a single trajectory x, with several variables (> 50)
- x is divided as x^+_{train}, x^+_{test} , x^-_{test}
- Sensor readings are numerical variables, whilst actuator readings are ternary non-ordinal variables

Results

	Multi-run G3P (30 runs)				One-shot G3P ($n_{target} = 20$)			
Dataset	TPR	FPR	AUC	f_{evals}	TPR	FPR	AUC	f_{evals}
SWaT	0.6648	0.0005	0.8321	43 243	0.6571	0.0007	**0.8401**	**11 767**
N-BaIoT-1	0.9981	0.0000	**0.9990**	47 152	0.8952	0.0011	0.9475	**3297**
N-BaIoT-2	0.9996	0.0016	0.9989	355 696	1.0000	0.0422	**0.9998**	**5732**
N-BaIoT-3	0.9949	0.0000	**0.9974**	51 979	0.9596	0.0076	0.9739	**5965**
N-BaIoT-4	0.0000	0.0002	0.4998	298 158	0.9272	0.0025	**0.9632**	**35 811**
N-BaIoT-5	0.6152	0.0012	0.8073	156 033	0.7492	0.0010	**0.8742**	**7898**
N-BaIoT-6	0.7192	0.0011	**0.8594**	371 358	0.6807	0.0023	0.8387	**12 235**
N-BaIoT-7	0.7070	0.0000	0.8534	269 708	0.6896	0.0009	**0.9072**	**16 736**
N-BaIoT-8	0.0000	0.0000	0.5000	1 015 286	0.4166	0.0027	**0.7050**	**88 921**
N-BaIoT-9	0.7812	0.0005	**0.8905**	260 259	0.7440	0.0011	0.8702	**13 696**

Results

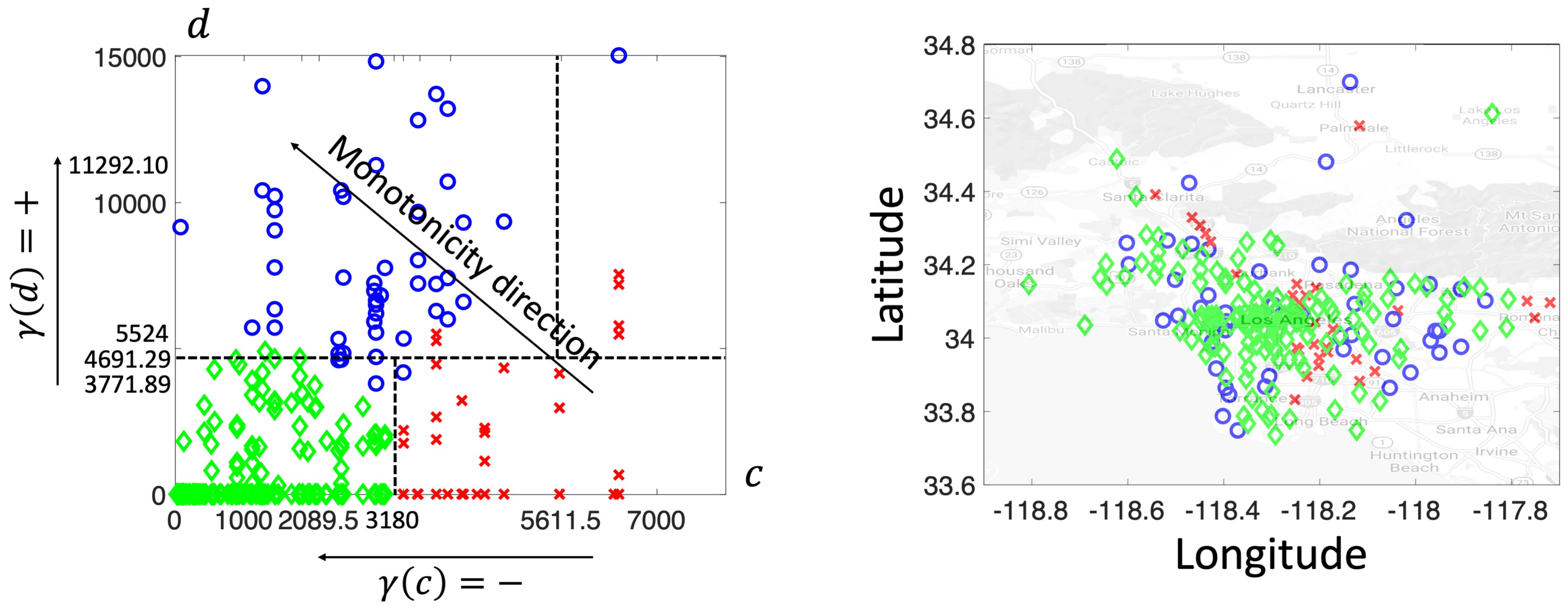
- Standard GP more than 60 % of the formulas containing a single variable.
- The one-shot algorithm produces a larger percentage of solutions with more variables, with some STL formulas containing more than 20 variables



Comparison with classical ML: it is

- competitive on SWaT
- it compares unfavourably on N-BaIoT, where it reaches a perfect detection rate only on N-BaIoT-2. However on N-BaIoT at least one anomalous instant for each attack is correctly identified, and all attacks might thus be considered as identified.

Learning STL-based clustering (Unsupervised Learning)



Goal: clusterizing spatio-temporal data using formal logic

[Mohammadinejad et al, Mining Interpretable Spatio-temporal Logic Properties for Spatially Distributed Systems, ATVA, 2021]

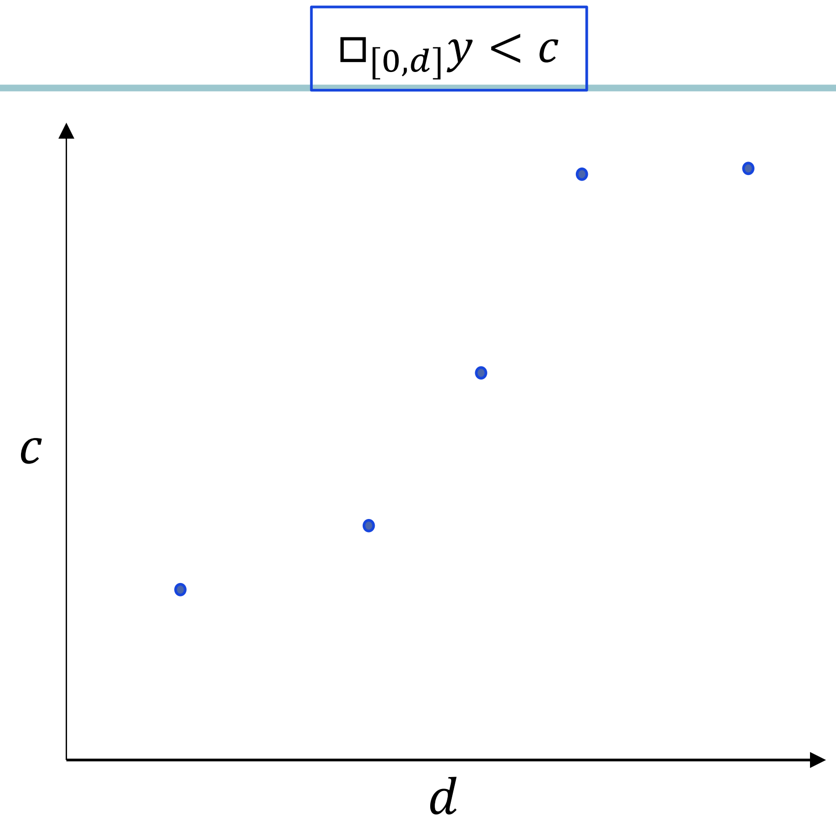
Monotonic PSTREL $\varphi(p)$:

- The polarity of a parameter p is:
  - $+$ if it is easier to satisfy φ as we increase the value of p
  - $-$ if it is easier to satisfy φ as we decrease the value of p
- Monotonic PSTREL:
  - All parameters have either $+$ or $-$ polarity
- Example: $\square_{[0,d]}\varphi$
  - Polarity of d is $-$

Validity Domain of PSTREL $\varphi(p)$

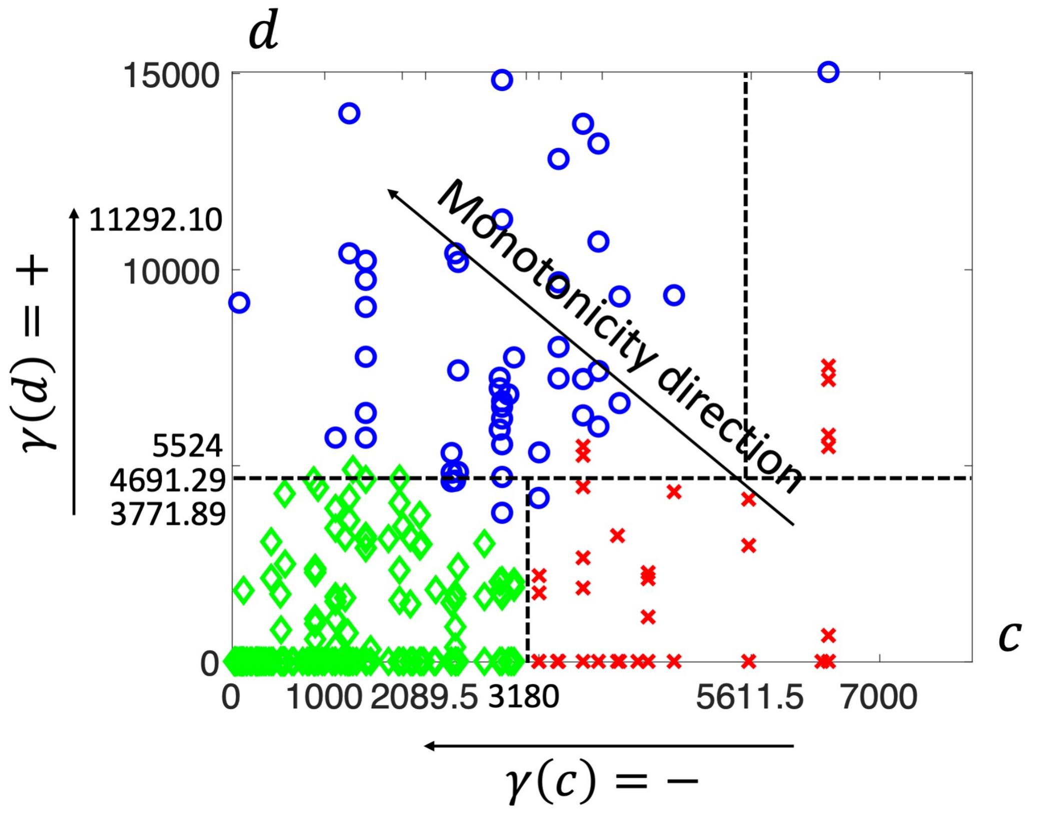
- Given a location l
- A set of spatio-temporal traces X associated with l
- The set of all valuations to p such that each trace in X satisfies the STREL formula
- Boundary of the validity domain:

  The robustness value with respect to at least one trace in X is ≈ 0



High-level steps

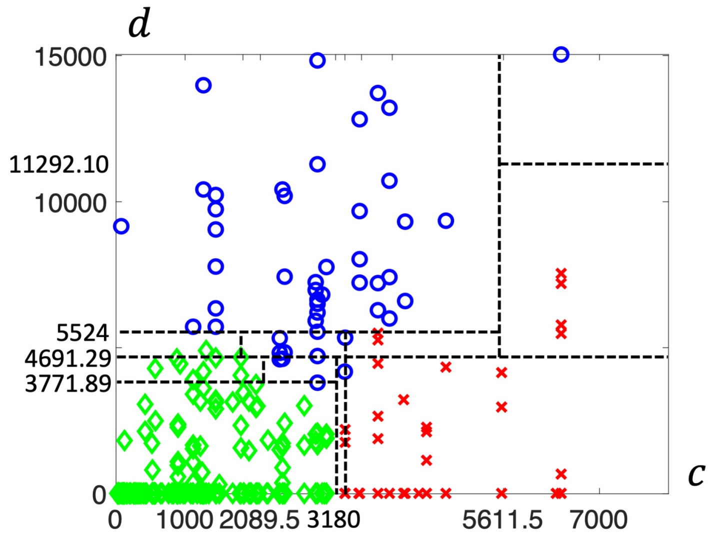
- Constructing the spatial model
- Projecting each spatio-temporal trace to a tight valuation in the parameter space of a given PSTREL formula
- Clustering the trace projections throught AHC
- Learning bounding boxes for each cluster using a Decision Tree based approach
- Learning a STREL formula for each cluster
- Improving the interpretability of the learned STREL formulas

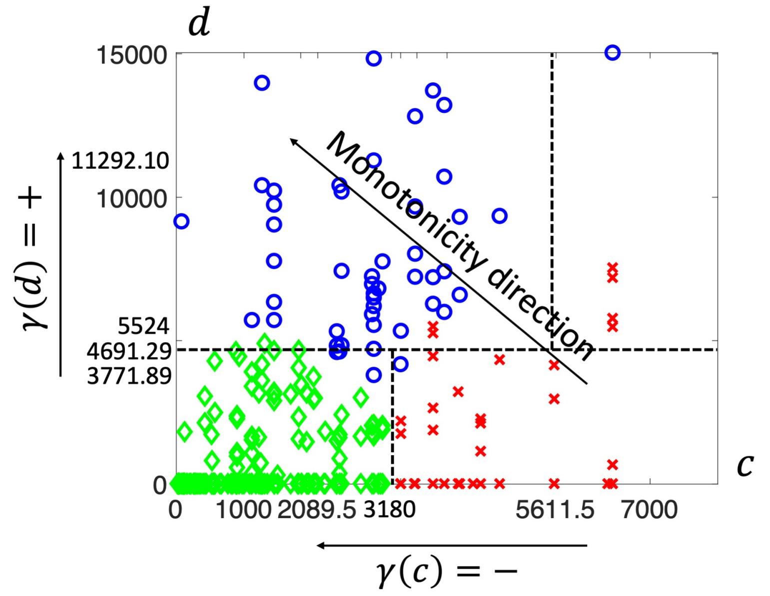


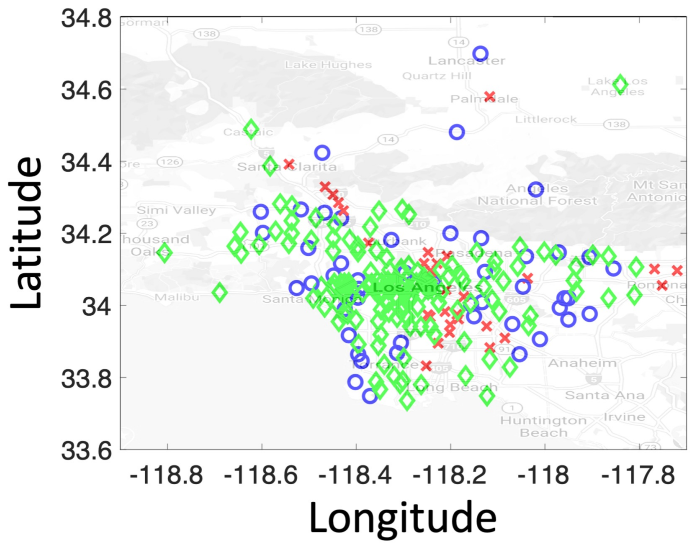
COVID-19 data from LA County

PSTREL formula: $\Diamond_{[0,d]}\{F_{[0,\tau]}(x > c)\}$

- We fix τ to 10 days
- Small d and large c are **hot spots**





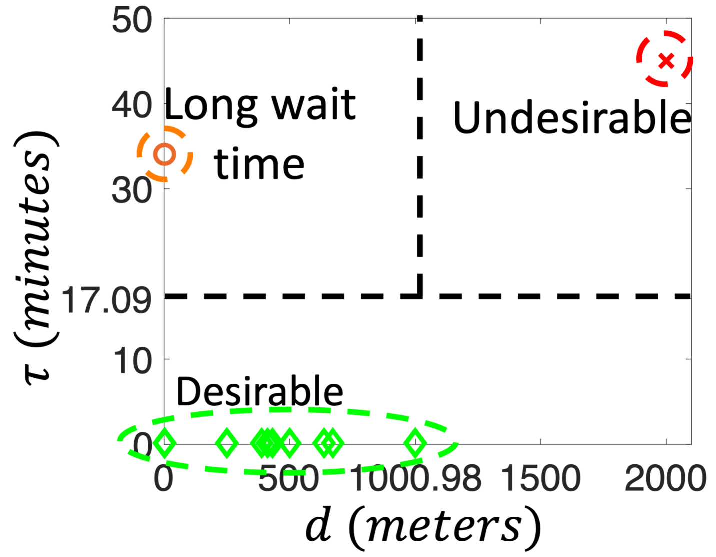


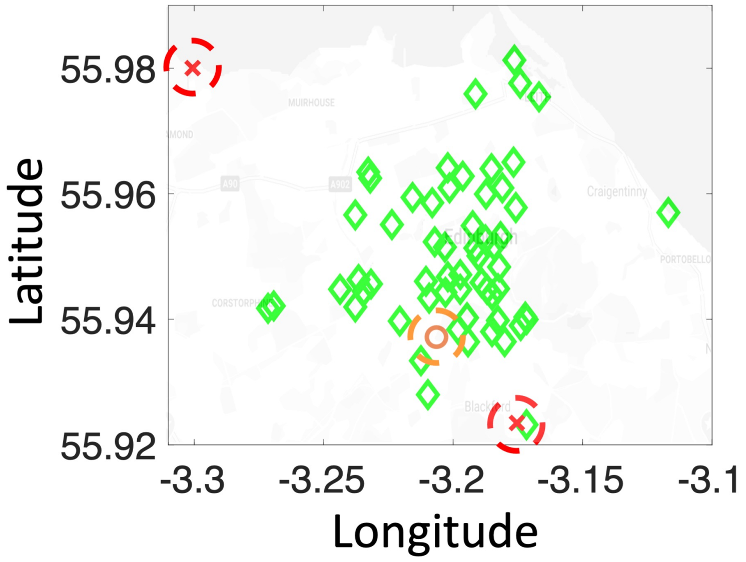
$$\varphi_{red} = \Diamond_{[0,4691.29]}\{F_{[0,10]}(x \geq 3181)\} \vee \Diamond_{[0,15000]}\{F_{[0,10]}(x \geq 5612)\}$$

BSS data from the city of Edinburgh

PSTREL formula: $\varphi(\tau, d) = G_{[0,3]}\left(\varphi_{wait}(\tau) \vee \varphi_{walk}(d)\right)$

$\varphi_{wait}(\tau) = F_{[0,\tau]}(B \geq 1) \wedge F_{[0,\tau]}(S \geq 1),$
$\varphi_{walk}(d) = \diamond_{[0,d]}\,(B \geq 1) \wedge \diamond_{[0,d]}\,(S \geq 1)$



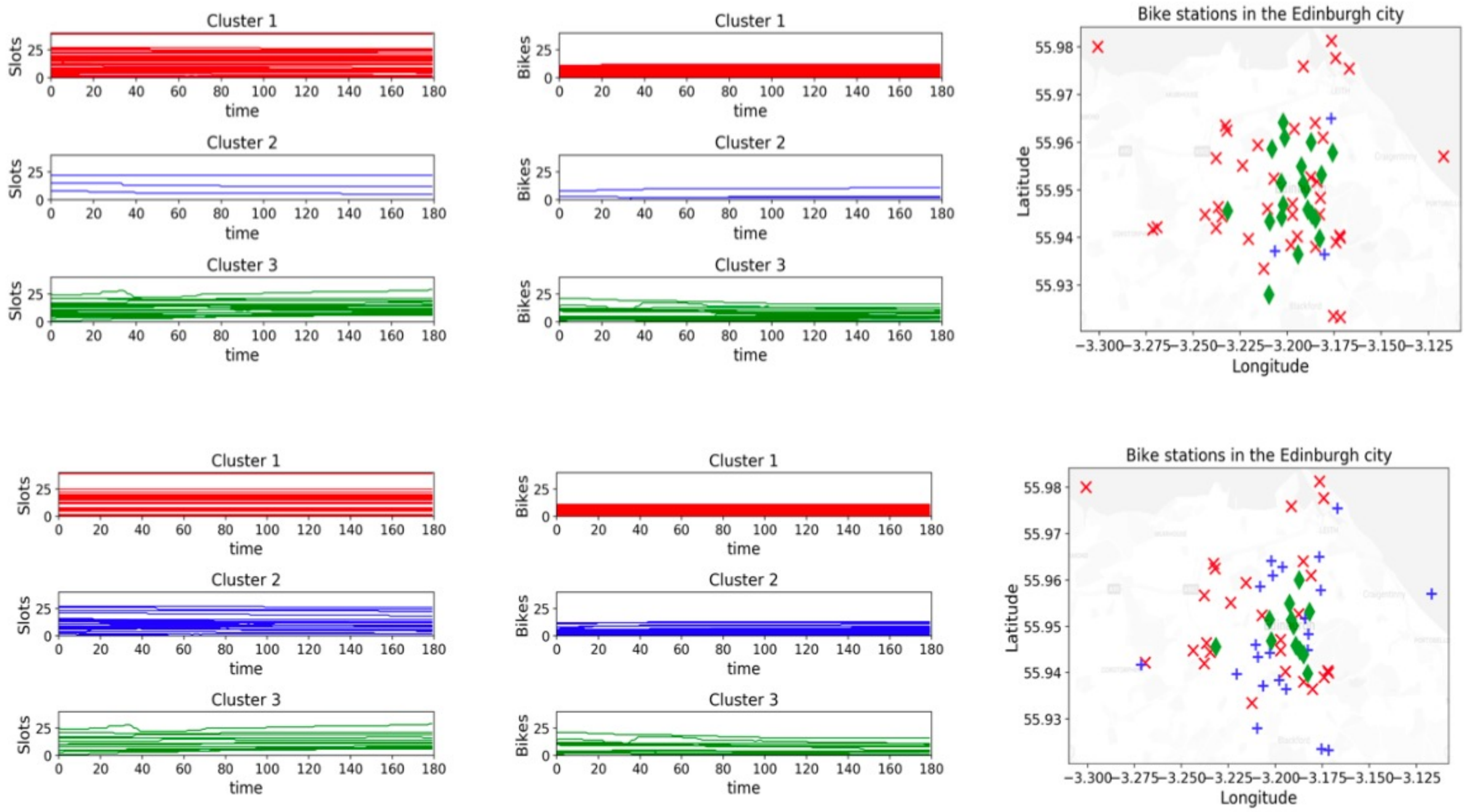


$$\varphi_{red} = \neg G_{[0,3]}\left(\varphi_{wait}(17.09) \vee \varphi_{walk}(2100)\right) \wedge \neg G_{[0,3]}(\varphi_{wait}(50) \vee \varphi_{walk}(1000.98))$$

Traditional ML approaches

Kshape approach from tslearn library

KMeans approach from tslearn library



(a) Clusters learned from BSS data

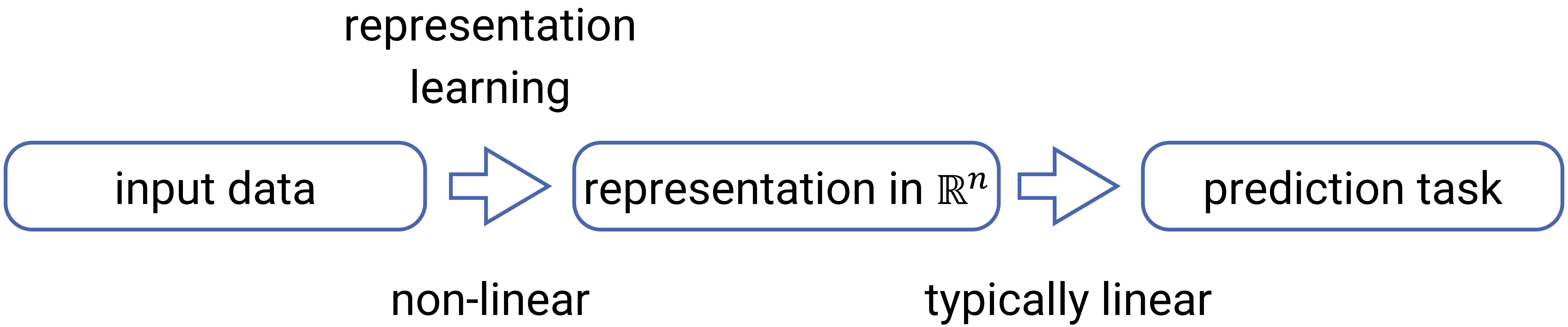
(b) Clusters learned from BSS data

(c) Regions in Edinburgh city associated with the learned clusters.

Related Works

- Bartocci et all: Survey on mining signal temporal logic specifications. Inf. Comput., 2022
- Template-Free:
  - Bombara, G et all, A Decision Tree Approach to Data Classification Using Signal Temporal Logic. In: Proc. of HSCC, 2016
  - Bombara, G. and Belta, C. (2021). Offline and Online Learning of Signal Temporal Logic Formulae Using Decision Trees.
  - Mohammadinejad, S., Deshmukh, J. V., Puranic, A. G., Vazquez-Chanlatte, M., and Donze ́, A. (2020b). Interpretable classification of time-series data using efficient enumerative techniques. Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control.
  - Andrea Brunello, Dario Della Monica, Angelo Montanari, Nicola Saccomanno, Andrea Urgolo: Monitors That Learn From Failures: Pairing STL and Genetic Programming. IEEE Access 11:
- Only-positive Example:
  - S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, N. Shankar. TeLEx: learning signal temporal logic from positive examples using tightness metric, Formal Methods in System Design
- Clustering
  - Marcell Vazquez-Chanlatte, Jyotirmoy V. Deshmukh, Xiaoqing Jin, Sanjit A. Seshia: Logical Clustering and Learning for Time-Series Data. CAV (1) 2017: 305-325
- Exploiting Monotonicity
  - Marcell Vazquez-Chanlatte, Shromona Ghosh, Jyotirmoy V. Deshmukh, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia: Time-Series Learning Using Monotonic Logical Properties. RV 2018: 389-405

A modern machine learning approach



Goal: embed STL formulae in $\mathbb{R}^n$ meaningfully.

Ideally: distance between embedded formulae should reflect semantic distance.

Main: semantic-preserving embeddings

How to construct meaningful embeddings?

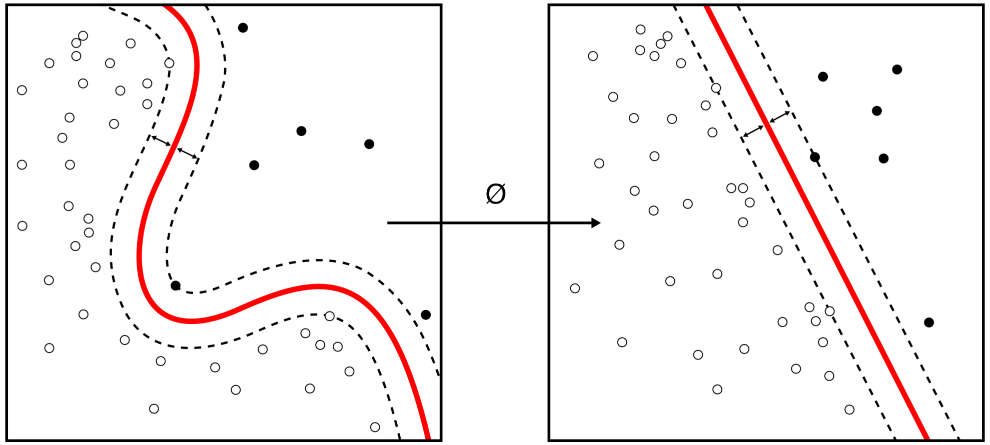
kernel-based methods

How to check that they are meaningful?

learning model checking

Bortolussi, L., Gallo, G. M., Křetínský, J., & Nenzi, L. *Learning model checking and the kernel trick for signal temporal logic on stochastic processes.* In: TACAS, 2022.

Kernels



Kernel's application to linearize a problem

A *kernel* is a function *k* defining implicitly a scalar product in a feature space

$$k(x, y) =< \phi(x), \phi(z) > \forall x, z \in X$$

where ϕ is a map from X to the feature space

Kernel Trick

A linear regression problem in the feature space $\phi(X)$: $\sum_j w_j \phi_j(x)$

has a dual formulation depending on N dual variables α and on the kernel evaluated among training points $k(x_i, x_j)$.

Overview: kernel trick for STL

1. How to embed formulae in a Hilbert space?

identify a formula with a functional via quantitative semantics: $\varphi : \mathcal{T} \rightarrow \mathbb{R}$

2. How to measure similarity on the feature representation?

use scalar product in L_2 w.r.t. a base finite measure μ_0

3. How to design a finite measure on trajectories?

prefer simple trajectories with limited variation

A kernel for STL

Computing kernels in three steps:

integration w.r.t. a base measure μ_0

$$k'(\varphi,\psi) = \int_{r \in \mathcal{T}} \varphi(r)\psi(r)d\mu_0(r)$$

normalisation

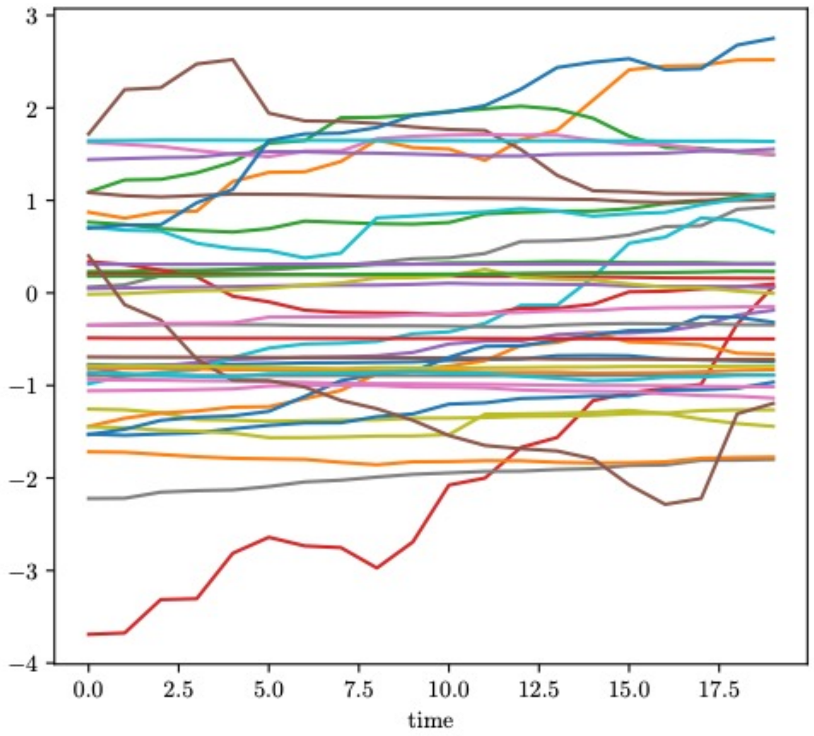
$$k_0(\varphi,\psi) = \frac{k'(\varphi,\psi)}{\sqrt{k'(\varphi,\varphi)k'(\psi,\psi)}}$$

exponentiation

$$k(\varphi,\psi) = \exp\left(-\frac{1-2k_0(\varphi,\psi)}{\sigma^2}\right)$$

The base measure

Compute integral by Montecarlo sampling of μ_0: $$k'(\varphi,\psi) \approx \frac{1}{M}\sum_{i=1}^{M}\varphi(r_i)\psi(r_i)$$



μ_0 is defined via its sampling algorithm:

- fixed time step Δ up to a final time T
- Bounded total variation (sampled from squared Gaussian)
- Limited change of sign of derivative

“Learning” model checking

Equipped with the previous definitions, we can try to solve the following problem:

Given $p(\psi_j|M)$ for **randomly** chosen formulae $\psi_1, \dots, \psi_n$

can we predict $p(\varphi|M)$?

without knowing or executing the system M

Learning with STL kernels

Different kinds of prediction tasks:

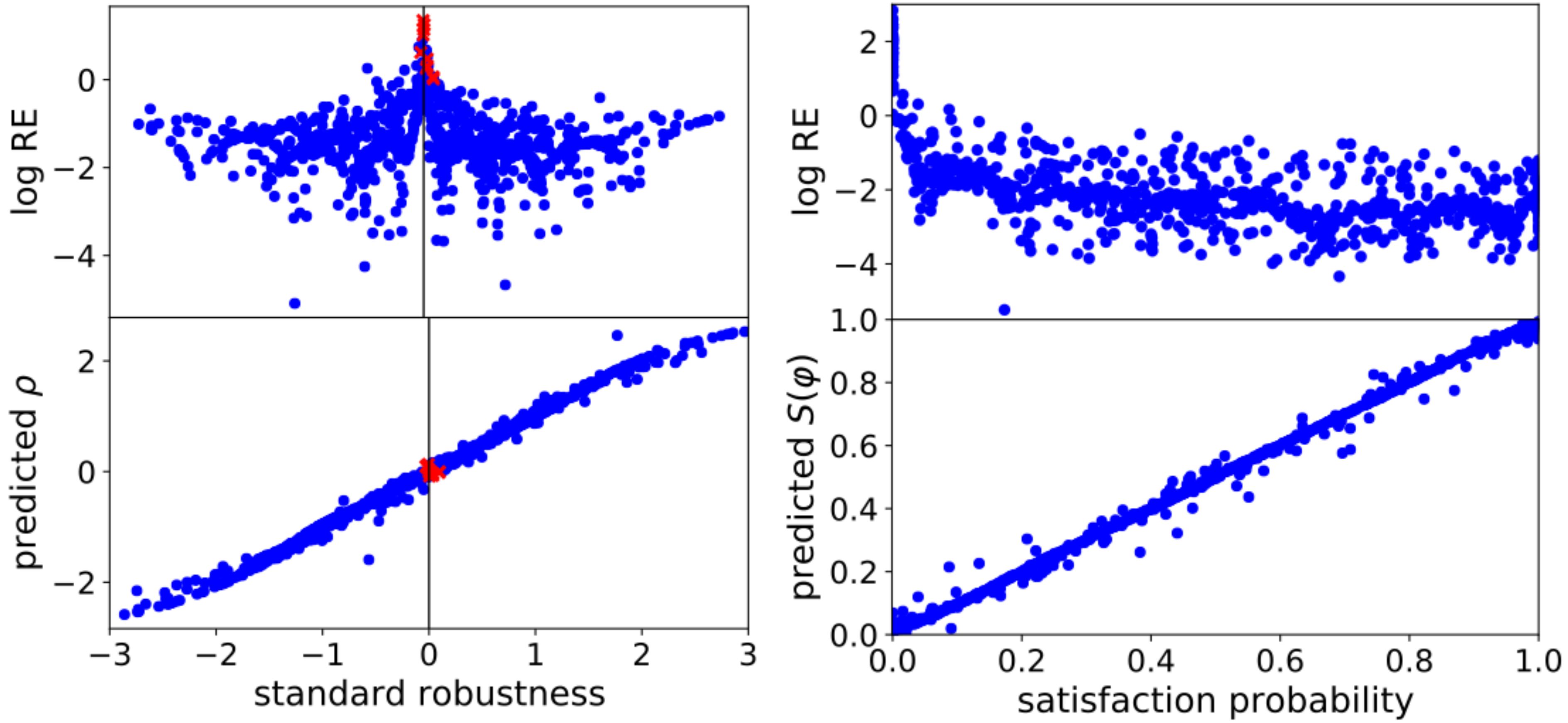
- Boolean truth and robustness for individual trajectories
- average robustness (w.r.t. μ_0 or a generic process μ)
- satisfaction probability (w.r.t. μ_0 or a generic process μ)

Data distribution over STL formulae φ: prefer simple formulae over complex ones

Training set: $\{(\psi_j, y_j)\}_{j=1\ldots,n}$

Learning algorithm: kernel ridge regression (with cross-validation)

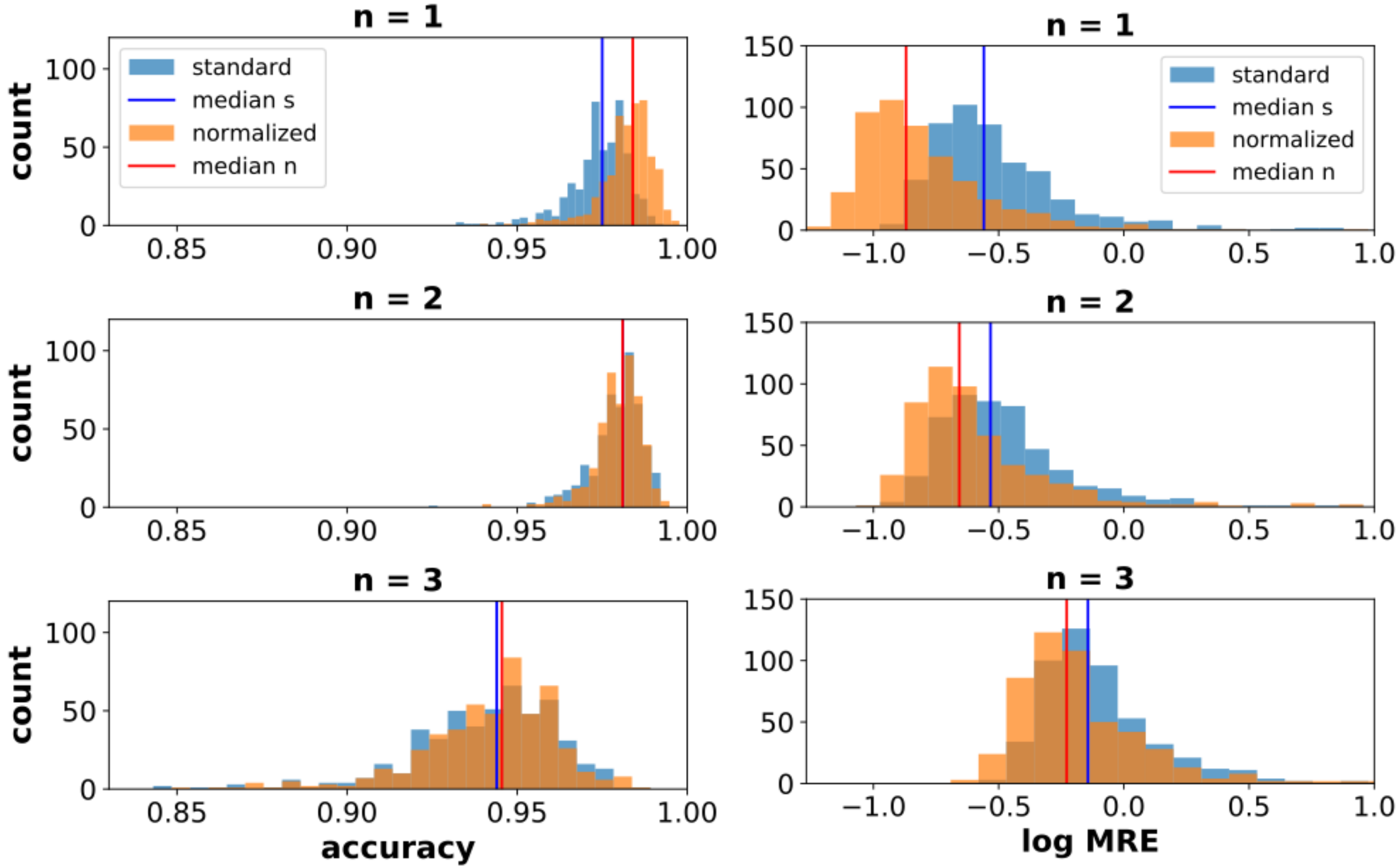
Experimental Results



(left) Robustness on single trajectories and (right) satisfaction probability (μ_0)

Good generalisation on out-of-distribution formulae

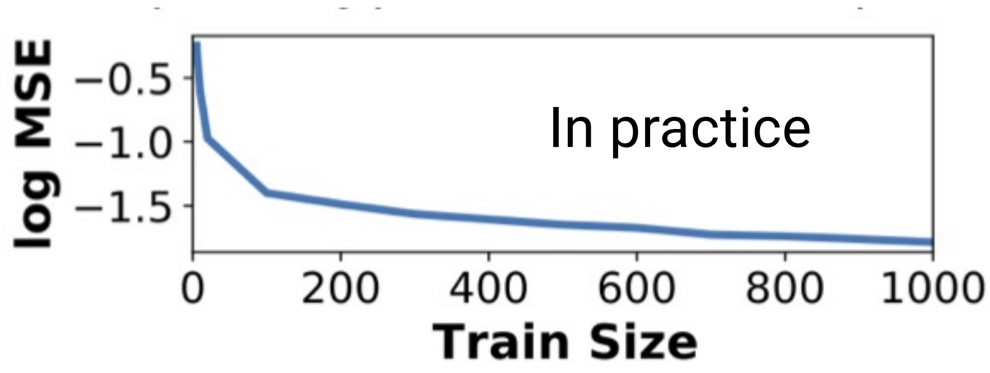
Experimental Results on the stochastic models



(left) Accuracy of satisfiability prediction and (right) MRE of robustness prediction

Immigration (1d)
Isomerization (2d)
Transcription (3d)

How many input points we need?



PAC bounds for 0-1 loss:

$$L(h) \leq \hat{L}_D(h) + \frac{\Lambda}{\sqrt{m}} + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}.$$

Λ: maximum norm of regression functions; δ: error probability; m: dataset size;

$$L(h) = \mathbb{E}_{\varphi \sim p_{data}}\left[\mathbb{I}\big(h(\varphi) \neq y(\varphi)\big)\right]; \quad \hat{L}_D(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}\big(h(\varphi_i) \neq y(\varphi_i)\big)$$

Dessert: ongoing work

How to make embeddings explicit (i.e. in $\mathbb{R}^k$)?

kernel PCA

Can we replace quantitative with Boolean semantics?

Boolean kernel

How to use these embeddings for STL requirement mining?

invert the embeddings using GNN

From implicit to explicit embeddings

input data → representation in $\mathbb{R}^\infty$ → representation in $\mathbb{R}^k$ → prediction task

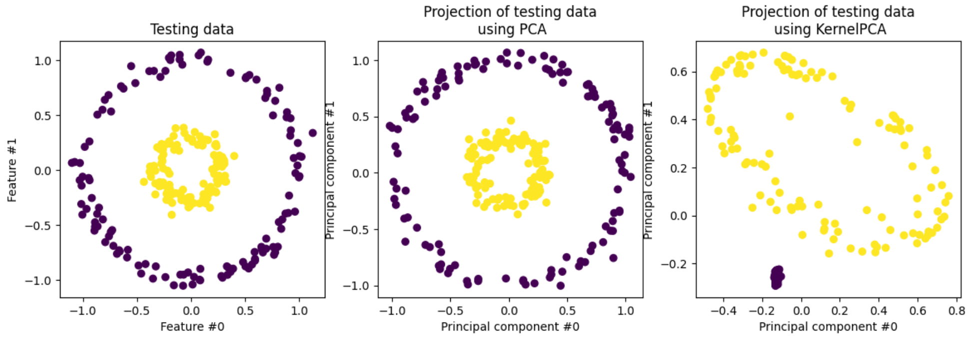
STL kernel

dimensionality reduction $k << n < \infty$

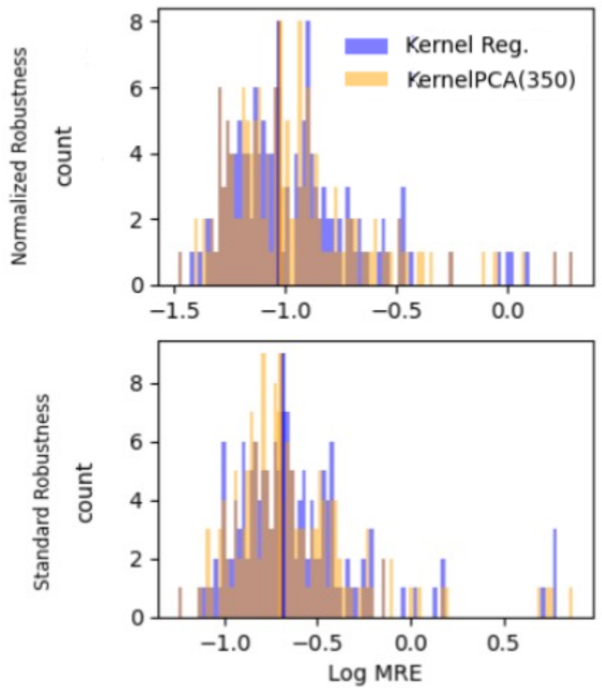
Goal: reduce the dimensionality of the embeddings using Kernel-PCA

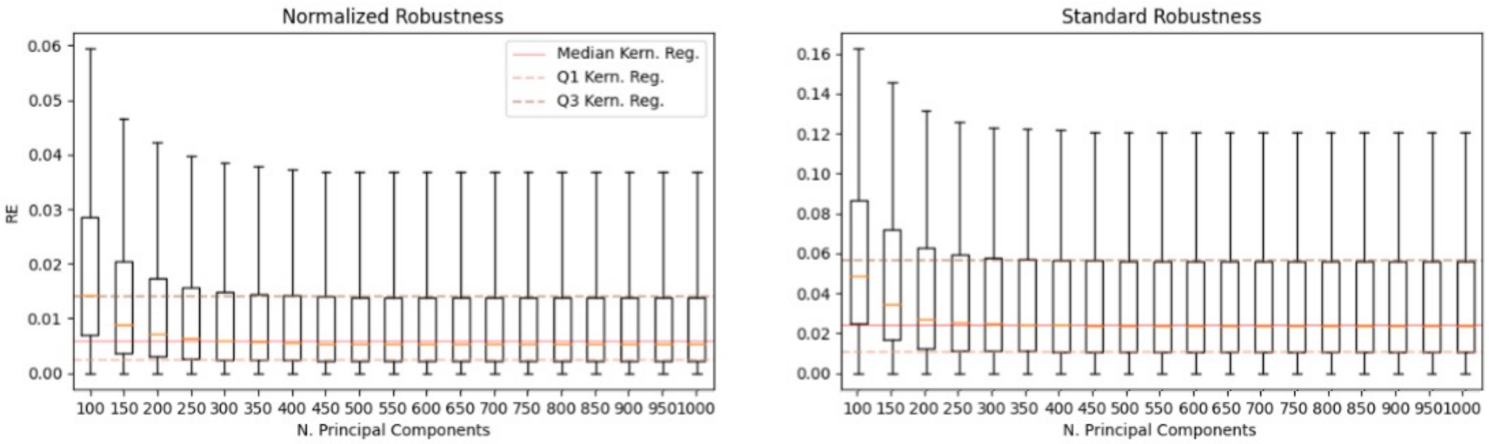
Kernel-PCA

Project input data on a high-dimensional continuous space $\mathbb{R}^n$ using a kernel, then perform dimensionality reduction using PCA to project the embeddings in $\mathbb{R}^k$, where downstream tasks are performed.



Kernel-PCA: experimental results





After ~ 350 principal components, the performance of Kernel PCA stabilises to errors comparable to that of STL Regression.

MRE Comparison of STL Kernel Regression with $n = 1000$ and Kernel PCA + linear regression with $k = 350$.

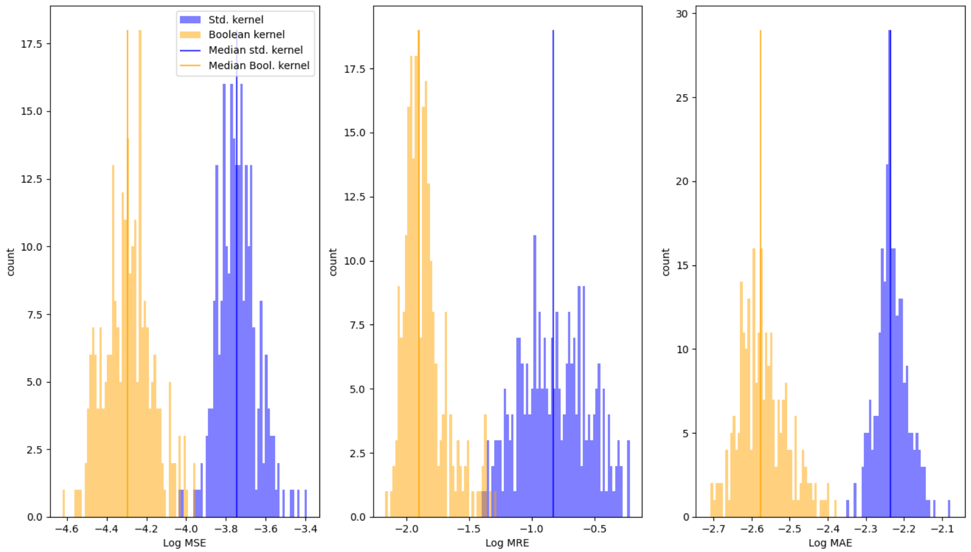
Intuition: many of the formulae in the training set bring the same contribution to the final predictions, without adding a significant amount of information. Reducing the dimension of the embeddings saves computational time without hurting the predictive performance.

A STL-kernel leveraging qualitative satisfaction

Adapt the definition of the STL Kernel to rely on the qualitative/Boolean semantics of STL

$$k'_b(\varphi,\psi) = \int_{r\in\mathcal{T}} \overline{\varphi}(r)\overline{\psi}(r)\mathrm{d}\mu_0(r)$$

i.e. integral of the product of the satisfiability value of input formulae w.r.t. measure μ_0.



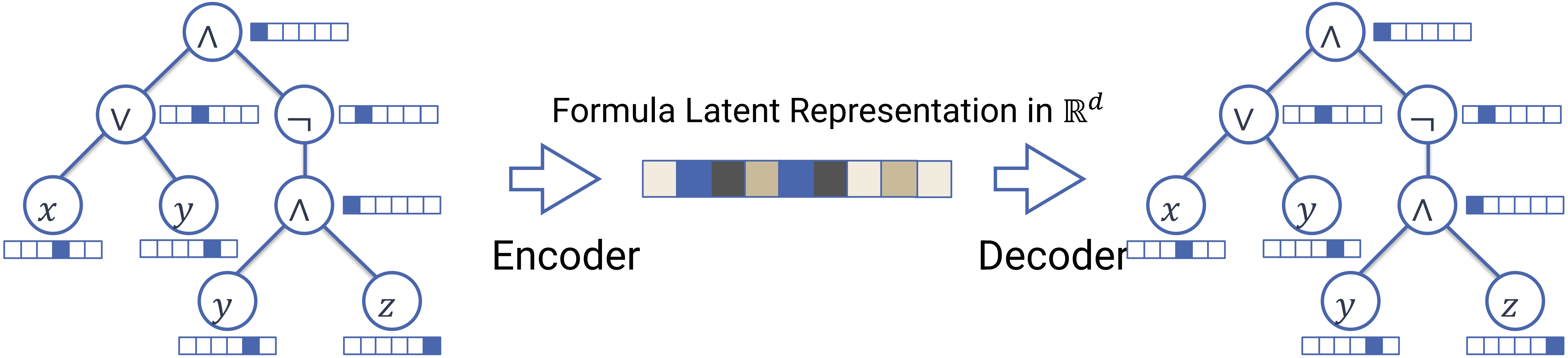
Satisfaction probability

Advantages:

- the Boolean kernel preserves semantic equivalence
- the Boolean kernel outperforms the standard one on the task of satisfaction probability;
- interpretable measure of similarity between STL formulae (allowing to sample formulae as diverse as possible).

Inverting the embedding

Problem with kernel embeddings: non-invertibility → **encoding-decoding architecture**



Learn invertible encodings using Graph Neural Networks (GNN):

- Encode parse tree of the formula into the latent space
- Decode latent vectors to syntactic trees, ideally with the same semantic meaning of the input formula

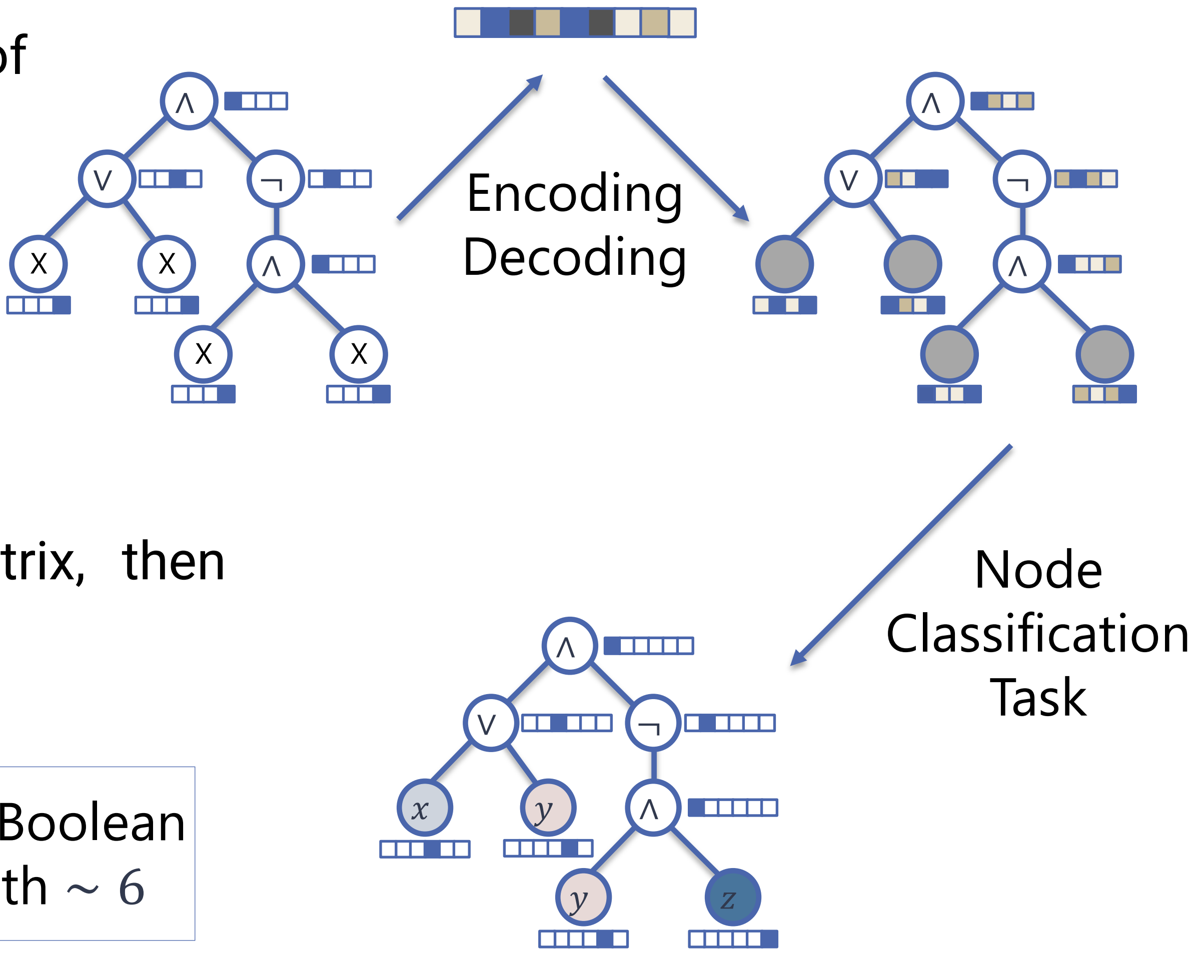
A simpler setting: boolean formulae

Problems with GNN encoding-decoding architectures:

- Scalability to deeper parse trees
- Learning temporal/threshold parameters of operators

Current solutions/attempts:

- Boolean logic setting (i.e. non-parametric formulae)
- Hierarchical approach: first learn adjacency matrix, then features



Currently $92 \pm 3\%$ average reconstruction accuracy on Boolean formulae with 5 variables and parse trees having depth ~ 6

Conclusions

- Using kernels + kernel PCA, we can construct finite dimensional embeddings which are effective in solving the “learning” model checking problem.
- Leveraging GNN deep learning models we are trying to build syntax based invertible embeddings.
- Idea: combine syntax and semantic based embeddings to get invertible mappings from formulae to real vector spaces
- use the framework for STL requirement mining, formula translation, sanitisation and simplification, game-based synthesis, …