TADAM: Learning Timed Automata from Noisy Observations

Lénaïg Cornanguer[1], Pierre-François Gimenez[2]
[1]CISPA Helmholtz Center for Information Security, [2]Inria
lenaig.cornanguer@cispa.de, pierre-francois.gimenez@inria.fr

SDM'25

Context: Automata Mining

Automata are:

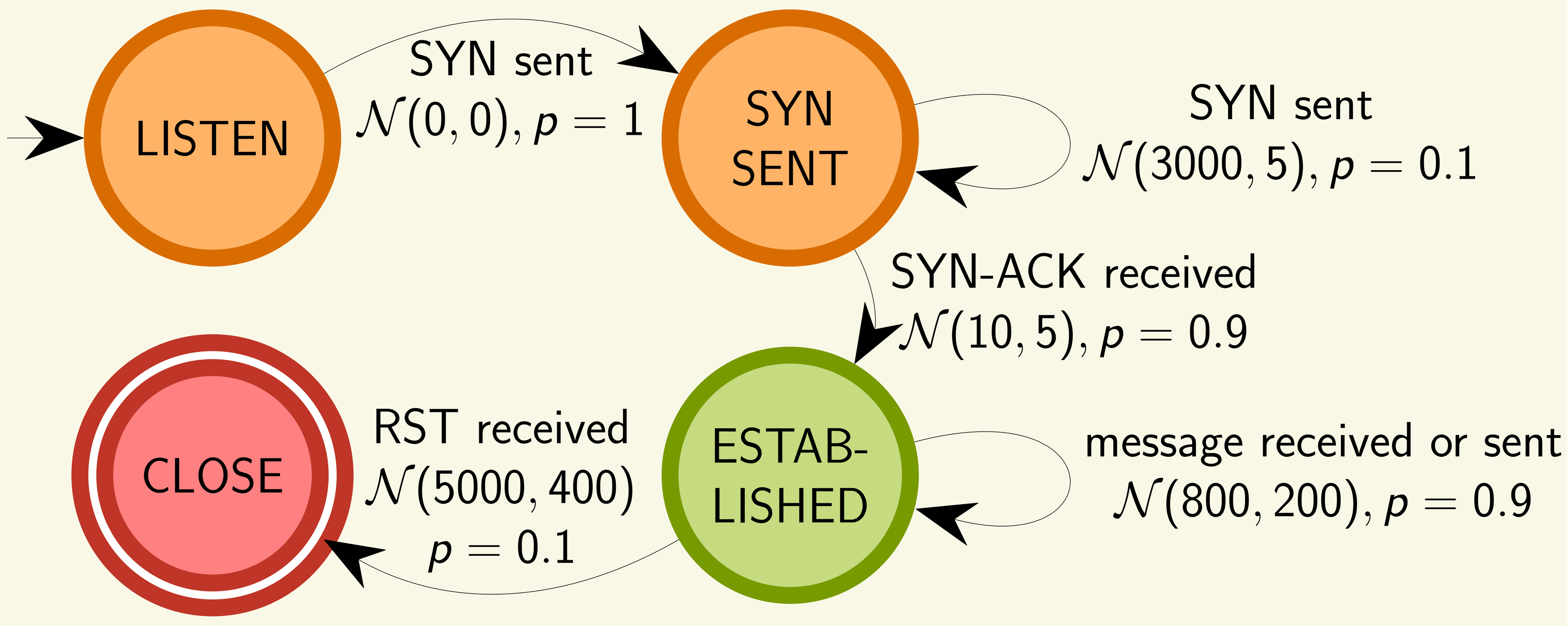
- Human-understandable
- Useful for monitoring, model checking, data generation...
- A natural fit for systems relying on finite-state machines

Probabilistic Real-Time Automata Language

Probabilistic helpful for identifying typical behavior; crucial for anomaly detection and data generation

Real-time modelize delays between events as distributions

We observe **events: (symbol, delay)**



Research Question

- We are interested in **passive learning** from **positive examples** only
- Limited measurement accuracy, configuration error, non-deterministic behavior can lead to **noisy observations**
- Related work cannot handle noisy observations

Research question: **how to learn probabilistic real-time automata from noisy observations?**

Noise Model

We propose an explicit modelization of the noise with:

- deletion of an event
- insertion of an event
- transposition of two events
- symbol repetition

Model Encoding

MDL principle: the best model compresses the data the most

$$L(\mathcal{A}) = L_{\mathbb{N}}(|\mathcal{Q}|) + L_{\mathbb{N}}(|\Sigma|) + \sum_{e \in \mathcal{E}} \Big(2\log_2(|\mathcal{Q}|) + \log_2(|\Sigma|) + L_{\mathbb{N}}(\lfloor \mu_e \rfloor) + L_{\mathbb{N}}(\lfloor \sigma_e^2 \rfloor)\Big) + 2\log_2(|\mathcal{Q}|)$$

It encodes: the location, the alphabet, the initial and accepting locations and the transitions

Data Encoding

We could encode data according to their probability but **noisy data would have null probability**!

Data encoding as a two-step process:

1. Correct non-accepted words to remove the noise
2. Encode the corrected data and their correction

For each noise type, there is a correction operation (deletion → add, etc.). Overall cost of the correction is minimized by a variation of the **Levenshtein distance algorithm**

Elementary Automaton Operations

Learning is based on three elementary operations:

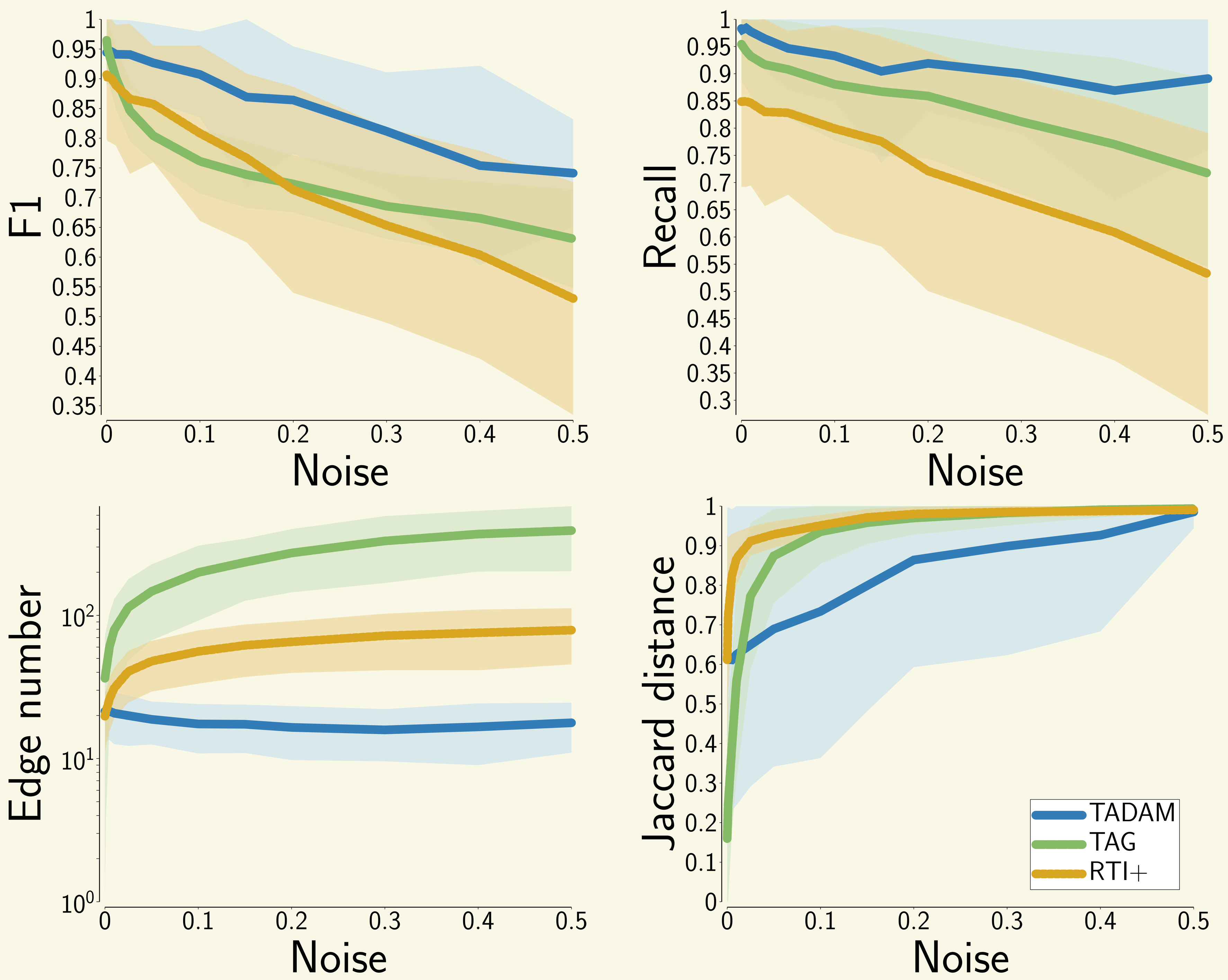
- **Location merge** (model cost ↘, data cost ↗)
- **Location split** (model cost ↗, data cost ↘)
- **Subpart deletion** (model cost ↘, data cost ↗)

TADAM Learning Algorithm

Data: Input sample of timed sequences $\mathcal{D}$

1 $\widehat{\mathcal{A}} \leftarrow$ MarkovInit($\mathcal{D}$)
2 **repeat**
3   *candidates* $\leftarrow \{\text{transform}(\widehat{\mathcal{A}}, \text{operation}, \text{target})\}$
4   $\mathcal{A}' \leftarrow \arg\min_{\mathcal{A} \in \text{candidates}} L(\mathcal{A}) + L(\mathcal{D}|\mathcal{A})$
5   $gain \leftarrow L(\widehat{\mathcal{A}}) + L(\mathcal{D}|\widehat{\mathcal{A}}) - L(\mathcal{A}') - L(\mathcal{D}|\mathcal{A}')$
6   **if** $gain > 0$ **then** $\widehat{\mathcal{A}} \leftarrow \mathcal{A}'$;
7 **until** $gain \leq 0$;
8 **return** $\widehat{\mathcal{A}}$

Noise Robustness on Synthetic Data



- TADAM is more robust to noise
- It learns smaller models that are easier to understand

Anomaly Detection in System Logs

Learner	AUROC	TPR	FPR	F1
TADAM	**0.982**	0.998	**0.025**	**0.705**
TAG	0.891	**1**	0.142	0.298
RTI+	0.790	**1**	0.292	0.171
HMM	0.608	0.640	0.085	0.288

- TADAM has very high detection rate and few false alarms
- TAG and RTI+ overfit on training data and do not generalize properly
- HMM is not expressive enough

Perspectives

- Extension to more complex automata languages:
  - timer automata
  - counter-based automata
  - pushdown automata
- Application to reverse engineering of undocumented network protocols

Test it!

Fos-R/TADAM

`pip install tadam-learner`