nature portfolio

https://doi.org/10.1038/s41562-022-01372-0

Supplementary information

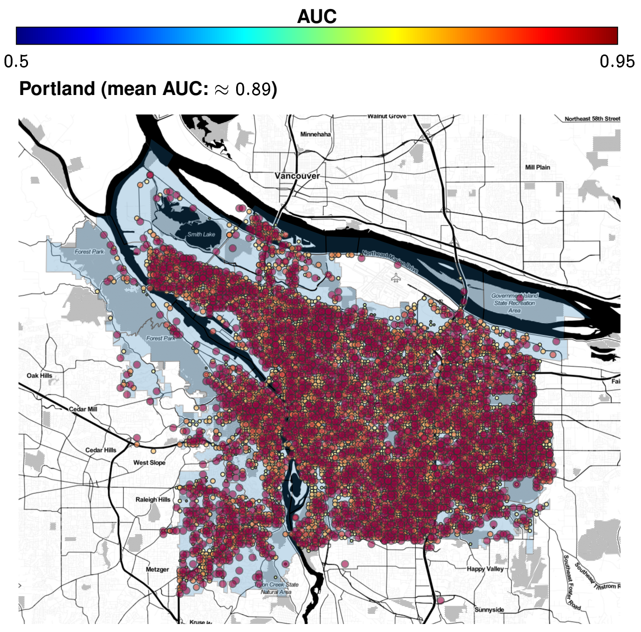
Event-level prediction of urban crime reveals a signature of enforcement bias in US cities

Supplementary Information: Event-level Prediction of Urban Crime Reveals Signature of Enforcement Bias in U.S. Cities

Yi Huang[1], Victor Rotaru[1,3], Timmy Li[1,3], James Evans[2,4,6] and Ishanu Chattopadhyay,[1,4,5]★

[1]Department of Medicine, University of Chicago, Chicago, IL 60637, USA
[2]Department of Sociology, University of Chicago, Chicago, IL 60637, USA
[3]Department of Computer Science, University of Chicago, Chicago, IL 60637, USA
[4]Committee on Quantitative Methods in Social, Behavioral, and Health Sciences, University of Chicago, Chicago, IL 60637, USA
[5]Committee on Genetics, Genomics & Systems Biology, University of Chicago, Chicago, IL 60637, USA
[6]Santa Fe Institute, Santa Fe NM 87501, USA

★To whom correspondence should be addressed: e-mail: `ishanu@uchicago.edu`.



Supplementary Figure 1. AUC distribution obtained for the City of Portland for the NIJ forecast challenge.

SUPPLEMENTARY METHODS

Algorithm Pseudocode

Algorithm 1: Granger Net

```
Data:
  • a set of sequence $\{x_i : i = 1, \dots, N\}$ of length $n$;
  • a hyperparameter $0 < \varepsilon < 1$;
  • a model inference length $n_0 < n$;
  • a maximal delay $\Delta_{\text{max}}$;
  • a threshold coefficient of causal dependence $\gamma_0$ for admissible models;
Result: A set of XPFSA models and a set of scalar weights for each target $r \in \{1, \dots, N\}$.
/* Infer models */
1  Let $\mathcal{M}_r = \emptyset$ be the set of admissible models for each target $r \in \{1, \dots, N\}$;
2  for each delay $\Delta = 1, \dots, \Delta_{max}$ do
3      for each source $s = 1, \dots, N$ and target $r = 1, \dots, N$ do
4          Let $x_{\text{in}} = (x_s)_1^{n_0 - \Delta}$;
5          Let $x_{\text{out}} = (x_r)_{\Delta+1}^{n_0}$;
6          Calculate PFSA $G = \mathbf{GenESeSS}\,(x_{\text{in}}, \varepsilon)$;
7          Calculate XPFSA $H^s_{r,\Delta} = \mathbf{xGenESeSS}\,(x_{\text{out}}, \varepsilon)$;
8          Let $\gamma^s_{r,\Delta} = \mathbf{coefCausalDependence}(G, H^s_{r,\Delta})$;
9          if $\gamma^s_{r,\Delta} \geq \gamma_0$ then
10             Let $\mathcal{M}_r = \mathcal{M}_r \cup \left\{H^s_{r,\Delta}\right\}$;
/* Learn scalar weights */
11 for each target $r = 1, \dots, N$ do
12     Let $I_r = \left\{(s, \Delta) : \text{ there is a model } H^s_{r,\Delta} \in \mathcal{M}_r\right\}$;
13     for each timestamp $t = 1, \dots, n - n_0$ do
14         Let $\mathbf{x}_t$ be a vector with index set $I_r$;
15         for each pair $(s, \Delta) \in I_r$ do
16             Let $x_{\text{in}}$ the length $l$ sub-sequence of $x_s$ that ends in the $(n_0 + t - \Delta)$-th entry;
17             Let the entry of $\mathbf{x}_t[s, \Delta] = \mathbf{predict}\,\left(H^s_{r,\Delta}, x_{\text{in}}\right)$;
18             Let $y_t = x_r[n_0 + t]$;
19     Let $X$ the matrix with the $t$-th row being $\mathbf{x}_t$;
20     Let $\mathbf{y}$ be the vector with the $t$-th entry being $y_t$;
21     Initialize a suitable regressor $\mathbf{Reg}$;
22     Get scalar weights $\mathbf{w}_r = \left(w^s_{r,\Delta}\right)_{(s,\Delta)\in I_r} = \mathbf{Reg}\,(X, \mathbf{y})$;
23 return $\{(\mathcal{M}_r, \mathbf{w}_r) : r = 1, \dots, N\}$;
```

Algorithm 2: `GenESeSS`

```
Data: A sequence $x$ over alphabet $\Sigma$, $0 < \varepsilon < 1$
Result: State set $Q$, transition map $\delta$, and transition probability $\widetilde{\pi}$
/* Step One: Approximate $\varepsilon$-synchronizing sequence */
1  Let $L = \lceil \log_{|\Sigma|} 1/\varepsilon \rceil$;
2  Calculate the derivative heap $\mathcal{D}^x_\varepsilon$ equaling $\left\{ \hat{\phi}^x_y \ : \ y \text{ is a sub-sequence of } x \text{ with } |y| \leq L \right\}$;
3  Let $\mathcal{C}$ be the convex hull of $D^x_\varepsilon$;
4  Select $x_0$ with $\hat{\phi}^x_{x_0}$ being a vertex of $\mathcal{C}$ and has the highest frequency in $x$;
/* Step Two: Identify transition structure */
5  Initialize $Q = \{q_0\}$;
6  Associate to $q_0$ the sequence identifier $x^{\text{id}}_{q_0} = x_0$ and the probability vector $d_{q_0} = \hat{\phi}^x_{x_0}$;
7  Let $\widetilde{Q}$ be the set of states that are just added and initialize it to be $Q$;
8  while $\widetilde{Q} \neq \emptyset$ do
9      Let $Q_{\text{new}} = \emptyset$ be the set of new states;
10     for $(q, \sigma) \in \widetilde{Q} \times \Sigma$ do
11         Let $x = x^{\text{id}}_q$ and $d = \hat{\phi}^x_{x\sigma}$;
12         if $||d - d_{q'}||_\infty < \varepsilon$ for some $q' \in Q$ then
13             Let $\delta(q, \sigma) = q'$;
14         else
15             Let $Q_{\text{new}} = Q_{\text{new}} \cup \{q_{\text{new}}\}$ and $Q = Q \cup \{q_{\text{new}}\}$;
16             Associate to $q_{\text{new}}$ the sequence identifier $x^{\text{id}}_{q_{\text{new}}} = x\sigma$ and the probability vector $d_{q_{\text{new}}} = d$;
17             Let $\delta(q, \sigma) = q_{\text{new}}$;
18     Let $\widetilde{Q} = Q_{\text{new}}$;
19 Take a strongly connected subgraph of the labeled directed graph defined by $Q$ and $\delta$, and denote the vertex set of
     the subgraph again by $Q$;
/* Step Three: Identify transition probability */
20 Initialize counter $N[q, \sigma]$ for each pair $(q, \sigma) \in Q \times \Sigma$;
21 Choose a random starting state $q \in Q$;
22 for $\sigma \in x$ do
23     Let $N[q, \sigma] = N[q, \sigma] + 1$;
24     Let $q = \delta(q, \sigma)$;
25 Let $\widetilde{\pi}(q) = [\![ (N[q, \sigma])_{\sigma \in \Sigma} ]\!]$;
26 return $Q$, $\delta$, $\widetilde{\pi}$;
```

Algorithm 3: `xGenESeSS`

```
   Data: A sequence $x_{\text{in}}$ over alphabet $\Sigma_{\text{in}}$, a sequence $x_{\text{out}}$ over alphabet $\Sigma_{\text{out}}$, and $0 < \varepsilon < 1$
   Result: State set $R$, transition map $\eta$, and output probability $\chi$
   /* Step One: Approximate $\varepsilon$-synchronizing sequence */
1  Let $L = \lceil \log_{|\Sigma_{\text{in}}|} 1/\varepsilon \rceil$;
2  Calculate cross derivative heap $\mathcal{D}_\varepsilon^{x_{\text{in}}, x_{\text{out}}}$ equaling $\left\{\hat{\phi}_y^{x_{\text{in}}, x_{\text{out}}} : y$ is a sub-sequence of $x_{\text{in}}$ with $|y| \leq L\right\}$;
3  Let $\mathcal{C}$ be the convex hull $\mathcal{D}_\varepsilon^{x_{\text{in}}, x_{\text{out}}}$;
4  Select $x_0$ with $\hat{\phi}_{x_0}^{x_{\text{in}}, x_{\text{out}}}$ being a vertex of $\mathcal{C}$ and has the highest frequency in $x$;
   /* Step Two: Identify transition structure */
5  Initialize $R = \{r_0\}$;
6  Associate to $r_0$ the sequence identifier $x_{r_0}^{\text{id}} = x_0$ and the probability vector $\chi(r_0) = \hat{\phi}_{x_0}^{x_{\text{in}}, x_{\text{out}}}$;
7  Let $\widetilde{R}$ be the set of states that are just added and initialize it to be $R$;
8  while $\widetilde{R} \neq \emptyset$ do
9      Let $R_{\text{new}} = \emptyset$ be the set of new states;
10     for $(r, \sigma) \in \widetilde{R} \times \Sigma_{in}$ do
11         Let $x = x_r^{\text{id}}$ and $d = \hat{\phi}_{x\sigma}^{x_{\text{in}}, x_{\text{out}}}$;
12         if $||d - \chi(r')||_\infty < \varepsilon$ for some $r' \in R$ then
13             Let $\eta(r, \sigma) = r'$;
14         else
15             Let $R_{\text{new}} = R_{\text{new}} \cup \{r_{\text{new}}\}$ and $R = R \cup \{r_{\text{new}}\}$;
16             Associate to $r_{\text{new}}$ the sequence identifier $x_{r_{\text{new}}}^{\text{id}} = x\sigma$ and the probability vector $\chi(r_{\text{new}}) = d$;
17             Let $\eta(r, \sigma) = r_{\text{new}}$;
18     Let $\widetilde{R} = R_{\text{new}}$;
19 Take a strongly connected subgraph of the labeled directed graph defined by $R$ and $\eta$, and denote the vertex set of
     the subgraph again by $R$;
   /* Step Three: Identify output probability */
20 Initialize counter $N[r, \tau]$ for each pair $(r, \tau) \in R \times \Sigma_{\text{out}}$;
21 Choose a random starting state $r \in R$;
22 for $i \in 1, \ldots, |x_{in}|$ do
23     Let $\sigma_i$ be the $i$-th symbol in $x_{\text{in}}$ and $\tau_i$ be the $i$-th symbol in $x_{\text{out}}$;
24     Let $N[r, \tau_i] = N[r, \tau_i] + 1$;
25     Let $r = \eta(r, \sigma_i)$;
26 Let $\chi(r) = \left[\!\left[ (N[r, \tau])_{\tau \in \Sigma_{\text{out}}} \right]\!\right]$;
27 return $R$, $\eta$, $\chi$;
```

Theory of Probabilistic Automata

Granger Net is assembled from local models which are, in general, crossed probabilistic automata (XPFSA).

The construction of a Granger Net consists of two steps: 1) local model generation and network pruning and 2) local model aggregation for comprehensive prediction. Event prediction is accomplished by aggregating these local activations via a local regressor. No global optimization of these aggregation function is acrried out.

The model generation step of Granger Net is accomplished by the algorithms **GenESeSS** (See Algorithm 2) and **xGenESeSS** (See Algorithm 3). **xGenESeSS** produces XPFSA models that captures how the history of a source process influences the future of a target process. The Granger Net construction is described in Algorithm 1, and takes as input a set $\{x_s : s \in S\}$ of length-n time series, hyperparameters ε and $n_0 < n$ for local model inference, $\Delta_{\max}$ for maximum time delay, and γ_0 for thresholding admissible models. For each target sequence x_r, Granger Net outputs a set of admissible models $\mathcal{M}_r$ with a scalar weight for each model in $\mathcal{M}_r$ via model inference and pruning (line 1-10) and training of the aggregation weights (line 11-22).

Step 1: Model inference and pruning

The Granger Net framework models the influence from a source time series x_s on a target time series x_r at a particular time delay Δ by an XPFSA $H^s_{r,\Delta}$ (line 7). Thus, we infer $|S|\Delta_{\max}$ XPFSA models for each x_r which yields $|S|^2\Delta_{\max}$ models in total. Since the number of XPFSA models increases quadratically with the number of time series and strength of the links may vary, pruning low-performing models early is important for parsimony. Granger Net rejects models by thresholding on the *coefficient of causal dependence* $\gamma^s_{r,\Delta}$ of model $H^s_{r,\Delta}$ (line 8), which measures the strength of dependence of the output sequence on the input one. More specifically, we have

$$\gamma^s_{r,\Delta} = 1 - \frac{\text{uncertainty of the next output in } x_r \text{ with observation of } x_s}{\text{uncertainty of the next output in } x_r} \tag{1}$$

γ can be evaluated from the *synchronous composition* of the PFSA that models the input process (line 6) and the XPFSA that models the causal influence. Granger Net retains the model $H^s_{r,\Delta}$ if and only if $\gamma^s_{r,\Delta}$ is greater than a pre-specified threshold γ_0. At the conclusion of Step 1, Granger Net returns an admissible set of models

$$\mathcal{M}_r = \left\{H^s_{r,\Delta} : \gamma^s_{r,\Delta} > \gamma_0\right\} \tag{2}$$

for each $r \in S$.

Step 2: Train linear weights

In this step, we integrate the local models in x_r's admissible set for forecasting events in x_r. To do this, Granger Net trains a linear coefficient $\omega^s_{r,\Delta}$ for each $H^s_{r,\Delta} \in \mathcal{M}_r$ (line 22) so that the final prediction for x_r at time step h is equal to

$$\sum_{H^s_{r,\Delta} \in \mathcal{M}_r} \omega^s_{t,\Delta} H^s_{r,\Delta}\left((x_s)^{h-\Delta}\right), \tag{3}$$

where $(x_s)^{h-\Delta}$ is the truncation of x_s at $h - \Delta$. To compute the coefficients, we solve a regression problem $\mathtt{Reg}(X, \mathbf{y})$ (line 22) for each $r \in S$ with the predictor variables being predictions $\mathbf{x}_t[s, \Delta]$ obtained by running each sequence $(x_s)^{n_0+t-\Delta}$ through $H^s_{r,\Delta}$ (line 17), and the outcome variable being $x_r[n_0 + t]$, value of x_r at time $n_0 + t$ (line 18). Hence, the X matrix is the $(n - n_0) \times |\mathcal{M}_r|$ matrix with the entry indexed by $t, (s, \Delta)$ given by $\mathbf{x}_t[s, \Delta]$ and $\mathbf{y}$, the $(n - n_0)$-dimensional vector with the entry indexed by t given by $x_r[n_0 + t]$. We can solve for the linear weights with any standard regressor.

Inference Algorithms

On line 6 and 7 of Algorithm 1, Granger Net calls subroutine **xGenESeSS**, which infers XPFSA as models of cross-dependencies between processes. Here, we establish the correctness of **GenESeSS**.

The inference algorithm for PFSA is called **GenESeSS** for Generator Extraction Using Self-similar Semantics. The PFSA model is based on the concept of the *causal state.* A dynamical system reaches the same causal state via distinct paths if the futures are statistically indistinguishable. More precisely, each process over an alphabet Σ of size m gives rise naturally to an m-ary tree with the nodes at level d being sequences of length d, and the edge from the node x to $x\sigma$, $\sigma \in \Sigma$, labeled by $Pr(\sigma|x)$ – the probability of observing σ as the next output after x. By the definition of causal state, if two subtrees are identical with respect to edge labels, then their roots are sequences that lead the system to the *same* causal state. Identifying all the roots of identical subtrees induces a *finite* automaton structure whose unique strongly connected component is the generating model of the process.

Definition 1 (Probabilistic Finite-State Automaton (PFSA))**.** *A PFSA G is a quadruple* $(Q, \Sigma, \delta, \widetilde{\pi})$*, where* Q *is a finite set,* Σ *is a finite alphabet,* $\delta : Q \times \Sigma \to \Sigma$ *is called the transition map, and* $\widetilde{\pi} : Q \to \mathbf{P}_\Sigma$*, where* $\mathbf{P}_\Sigma$ *is the space of probability distributions over* Σ*, is called the transition probability.*

Step 2 of Algorithm 2 (line 5-19) is an implementation this subtree “stitching” approach under finiteness of input data.

Note that the criterion for "stitching" two subtrees with roots x and x' is that their edge labels are identical for *all* depths, which translates to $p(y|x) = p(y|x')$ for sequence y of all lengths. The criterion is not verifiable with finite data, and hence **`GenESeSS`** identifies two subtrees if they agree on depth one. Defining *symbolic derivative* ϕ_x to be the vector with the entry indexed by σ given by $p(\sigma|x)$, **`GenESeSS`** identifies x and x' if $\phi_x = \phi_{x'}$. This approach works well under the assumption that the target PFSA is in *general position*, meaning that different causal states have distinct symbolic derivatives. In practice, **`GenESeSS`** uses *empirical symbolic derivative* defined below to approximate ϕ_x. Let x be an input sequence of finite length, the empirical symbolic derivative $\hat{\phi}^x_y$ of a sub-sequence y of x is a probability vector with the entry indexed by σ given by

$$\hat{\phi}^x_y(\sigma) = \frac{\text{number of } y\sigma \text{ in } x}{\text{number of } y \text{ in } x} \tag{4}$$

`GenESeSS` identifies two sequences (line 12) if their *empirical* symbolic derivatives are within an ε-neighborhood of each other for certain $\varepsilon > 0$.

For simplicity, we first illustrate how **`GenESeSS`** solves the transition structure of the target PFSA from a sample path x generated from a process of Markov order k. Assuming the x_0 produced by Step 1 (line 4) is λ, the empty sequence, **`GenESeSS`** starts by calculating $\hat{\phi}^x_\lambda$, $i.e.$, the empirical distribution on Σ, and records λ as the identifier of the first state. Then, **`GenESeSS`** appends λ with each $\sigma \in \Sigma$, and calculates $\hat{\phi}^x_\sigma$. By the general position assumption and assuming x is long enough, with high probability, no $\hat{\phi}^x_\sigma$ is within an ε-neighborhood of $\hat{\phi}^x_{\sigma'}$ for $\sigma \neq \sigma'$, and hence each σ is recorded as the identifier for a new state. In fact, **`GenESeSS`** will keep on appending symbols to identifiers of stored states and adding new states until it reaches a sequence of length $k+1$. Assuming $y = \sigma_1 \cdots \sigma_k\sigma_{k+1}$, since the process is of order k, we have $\phi_y = \phi_z$ for $z = \sigma_2 \cdots \sigma_{k+1}$, and hence, with high probability, $\hat{\phi}^x_y$ and $\hat{\phi}^x_z$ can be within an ε-neighborhood of each other given long enough input x. In this case, **`GenESeSS`** identifies the state represented by y with that of z. In fact, **`GenESeSS`** will identify all states represented by sequences of length $k+1$ to some previously-stored states. And since no new states can be found, **`GenESeSS`** exits the loop on line 8 after iteration $k+1$. Taking the strongly connected component on line 19, **`GenESeSS`** gets the correct transition structure.

However, not all processes generated by PFSA have finite Markov order. For such cases, Step 2 of **`GenESeSS`** will never exit in theory, since there exists no $n \in \mathbb{N}$ such that every causal state is visited for sequences with length $\leq n$. And if we implement an artificial exit criterion, the model inferred might be unnecessarily large, and have hard-to-model approximations. We address this issue via the notion of synchronization – the ability to identify that we are localized or synchronized to a particular state despite being uncertain of the initial state.

In Step 1 of Algorithm 2 (line 1-4), **`GenESeSS`** finds an almost synchronizing sequence, which allows **`GenESeSS`** to distill a structure that is similar to that of the finite Markov order cases, and thus carry out the subtree "stitching" procedure described before. A sequence x is *synchronizing* if *all* sequences that end with the suffix x terminates on the same causal state. A process is synchronizable if it has a synchronizing sequence, and a PFSA is *synchronizable* if the process it generates is synchronizable. The structure of the "graph" of a perfectly synchronizable PFSA is that of a co-final automata[1].

A sequence x is *ε-synchronizing*[2] to the state q if the distribution $\wp_x$ on the state set Q induced by x satisfies $\|\wp_x - \mathbf{e}_q\|_\infty < \varepsilon$, where $\mathbf{e}_q$ is the base vector with 1 on the entry indexed by q and 0 elsewhere. The importance of ε-synchronizing sequence is twofold: 1) since $\phi^T_x = \wp^T_x\widetilde{\Pi}$, where $\widetilde{\Pi}$ is the $|Q| \times |\Sigma|$ matrix with the row indexed by q given by $\widetilde{\pi}(q)$, a $\wp_x$ close to $\mathbf{e}_q$ give rise to a ϕ_x close to $\widetilde{\pi}(q)$. And 2) although sequences prefixed by an ε-synchronizing sequence to a state q may not remain ε-synchronizing to state q, they are close to q *on average*.

To find an almost synchronizing sequence algorithmically[2], **`GenESeSS`** first calculates the convex hull of symbolic derivatives of subsequences of x up to length L (line 1-3), and then selects a sequence x_0 whose symbolic derivative is a vertex of the convex hull (line 4). Since the convex hull of $\left\{\phi_x : x \in \Sigma^L\right\}$ is a linear projection of the convex hull $\left\{\wp_G(x) : x \in \Sigma^L\right\}$ via $\widetilde{\Pi}$, we can expect sequence x with ϕ_x being a vertex of the convex hull of $\left\{\phi_x : x \in \Sigma^L\right\}$ to be a good candidate for an almost synchronizing sequence.

The corresponding inference algorithm for XPFSA is called **`xGenESeSS`**, which takes as input two sequences x_{in}, x_{out}, and a hyperparameter ε, and outputs an XPFSA in a manner very similar to the inference algorithm of PFSA.

While a PFSA models how the past of a time series influences its own future, a XPFSA models how the past of an input time series influences the future of an output time series. Hence, while in the SSC algorithm of PFSA, we identify sequences if they lead to futures that are statistically indistinguishable, in the SSC algorithm of XPFSA, we identify sequences if they lead to the same future distribution of the *output*.

Definition 2 (Crossed Probabilistic Finite-State Automaton (XPFSA))**.** *A crossed probabilistic finite-state automaton is specified by a quintuple* $(\Sigma_{in}, R, \eta; \Sigma_{out}, \chi)$*, where* Σ_{in} *is a finite input alphabet,* R *is a finite state set,* η *is a partial function from* $R \times \Sigma_{in}$ *to* R *called transition map,* Σ_{out} *is a finite output alphabet, and* χ *is a function from* R *to* $\mathbf{P}_{\Sigma_{out}}$ *called output probability map, where* $\mathbf{P}_{\Sigma_{out}}$ *is the space of probability distributions over* Σ_{out}*. In particular,* $\chi(r, \tau)$ *is the probability of generating* $\tau \in \Sigma_{out}$ *from a state* $r \in R$*.*

Note that a XPFSA has no transition probabilities defined between states as a PFSA does. The XPFSA in the example has a binary input alphabet and an output alphabet of size 3. The bar charts next to the 4 states of the XPFSA indicate the output probability distributions. To generate a sample path, an XPFSA requires an input sequence over its input alphabet.

Similar to the PFSA construction approach, here we compute the *cross symbolic derivative*, which is the ordered tuple $Pr(\tau|x)$, with $\tau \in \Sigma_{\text{out}}$ and a sequence x over Σ_{in}. We compute the empirical approximation of the cross symbolic derivative from sequences x_{in} and x_{out} as:

$$\hat{\phi}_y^{x_{\text{in}},x_{\text{out}}}(\tau) = \frac{\text{number of } \tau \text{ in } x_{\text{out}} \text{ after } y \text{ transpires in } x_{\text{in}}}{\text{number of sub-sequence } y \text{ in } x_{\text{in}}} \tag{5}$$

Thus, **`xGenESeSS`** is almost identical to **`GenESeSS`** except that, in Step 1, **`xGenESeSS`** finds an almost synchronizing sequence based on cross symbolic derivatives, and in Step 2, identifies the transition structure based on the similarity between cross symbolic derivatives. Arguments for establishing the effectiveness of **`GenESeSS`** carry over to **`xGenESeSS`** with empirical symbolic derivative replaced by empirical cross symbolic derivative.

Software for the cynet implementation, with instructions for installation and quick-start examples, is available at https://pypi.org/project/cynet/

References

[1] Ito, M. & Duske, J. On cofinal and definite automata. *Acta Cybernetica* **6**, 181–189 (1983).
[2] Chattopadhyay, I. & Lipson, H. Abductive learning of quantized stochastic processes with probabilistic finite automata. *Philos Trans A* **371**, 20110543 (2013).