A Related Work

Our work adds the interpretability to temporal point process models and extends the logic learning method.

Temporal point process models have been an elegant tool for event data learning, either for future prediction or (quasi-) causality discovery (Yan et al., 2019; 2016; Liu et al., 2017; Wu et al., 2018b). Traditional parametric models such as Hawkes process (Hawkes, 1971b) are built on simple assumptions such that past events will boost the occurrence of future events. To capture more complicated dynamics, Du et al. (2016a) proposed the first neural point process (NPP) model, where the intensity function is modeled by a Recurrent Neural Network (RNN). Mei & Eisner (2016) improved RMTPP by constructing a continuous-time RNN. Zuo et al. (2020); Zhang et al. (2020a) further leveraged the self-attention mechanism to capture the long-term dependencies of events and meanwhile enhance the computational efficiency. While these neural-based point process models are flexible and excel at event prediction, they are hard to interpret. To add transparency to the black-box models, recently Zhang et al. (2021) introduced Granger causality as a latent graph to explain point processes and the structures are jointly learned via gradient descent. However, Granger causality is still limited to the mutual triggering patterns of events. To incorporate more sophisticated and interpretable event patterns Li et al. (2020) proposed a Temporal Logic Point Process (TLPP), which constructs intensity function by pre-specified temporal logic rules. In fact, such temporal logic rule guided symbolic reasoning process will require highly expressive models such as Transformer to have a good approximate to their solutions (Finkbeiner et al., 2020). However, TLPP can not automatically discover rules and our TELLER overcomes this major drawback.

Probabilistic logic model dates back to Markov Logic Networks (MLN) (Richardson & Domingos, 2006; Singla & Domingos, 2008), where a Markov random field is used to model the first-order logic rules and their weights will be learned. Enhanced methods (Papai et al., 2012; Tran & Davis, 2008; Brendel et al., 2011) are generally based on probabilistic graphical models, which allow for partial and noisy observations yet at the cost of intensive computation for inference. TLPP (Li et al., 2020) and our TELLER simplify their setting with two assumptions, i.e. causal direction and fully observed states, and then the temporal logic model can be formulated as a continuous-time point process, which bridges the first-order temporal logic rules to point process.

Logic learning methods include SATNet (Wang et al., 2019a), which transforms rule mining into an SDP-relaxed MaxSAT problem, and attention-based methods (Yang & Song, 2019). Some works (Wang et al., 2017; Evans & Grefenstette, 2018; Wei et al., 2019) formulated logic learning as learning an explanatory binary classifier. For example, Evans & Grefenstette (2018) constructed a differentiable model that predicts the conditional probability of the outcome label for a ground atom. Neural-LP (Yang et al., 2017) provided the first fully differentiable rule mining method based on TensorLog (Cohen, 2016), and Wang et al. (2019b) extended Neural-LP to learn rules with numerical values via dynamic programming and cumulative sum operations. In addition, DRUM (Sadeghian et al., 2019) connected learning rule confidence scores with low-rank tensor approximation. Among all these methods, Dash et al. (2018) and Wei et al. (2019) first introduced the Column Generation algorithm in logic learning, which is the closest to our learning framework. However, all these methods are restricted to the static setting and cannot be directly implemented on event sequences.

B Allen's Thirteen Temporal Relation

Allen's original paper (Allen, 1990) defined 13 types of *temporal relations between two time intervals*. Specifically, define the time intervals for predicate x_A and predicate x_B as $I_{t_A} = (t_{A_1}, t_{A_2}]$ and $I_{t_B} = (t_{B_1}, t_{B_2}]$ respectively, where t_{A_1} and t_{B_1} are the transition times that the predicate enters the state, and and t_{A_2} and t_{B_2} are the time that the predicate leaves the state. The temporal relation, denoted as $R_{A\ type\ B}(I_{t_A}, I_{t_B})$, is a logic function defined over the time intervals.

See below Table 5 for an illustration. As shown in this table, the temporal relation can be mathematically evaluated by a step function

$$g(s) := \mathbb{1}(s \geq 0)$$

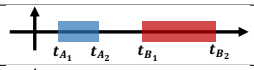
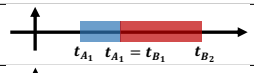
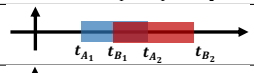
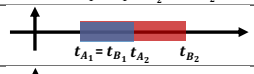
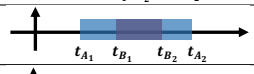
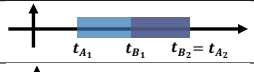
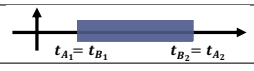
and an indicator function

$$\kappa(s) := \mathbb{1}(s = 0).$$

Considering the inverses of the listed relations plus the symmetric relation "Equal", there are a total of 13 relations.

In practice, to tolerate noise, i.e., the imprecisely recorded time information, it makes sense to introduce softened approximation functions for the step function $g(s)$ and the delta function $\kappa(s)$ in

Table 5: Interval-based temporal relations.

Temporal Relation	Logic Function $R_{A\ type\ B}(I_{t_A}, I_{t_B})$	Illustration
A Before B	$g(t_{B_1} - t_{A_2})$	
A Meets B	$\kappa(t_{A_2} - t_{B_1})$	
A Overlaps B	$g(t_{B_1} - t_{A_1}) \cdot g(t_{B_1} - t_{A_2}) \cdot g(t_{B_2} - t_{A_2})$	
A Starts B	$\kappa(t_{A_1} - t_{B_1}) \cdot g(t_{B_2} - t_{A_2})$	
A Contains B	$g(t_{B_1} - t_{A_1}) \cdot g(t_{A_2} - t_{B_2})$	
A Finished-by B	$g(t_{B_1} - t_{A_1}) \cdot \kappa(t_{A_2} - t_{B_2})$	
A Equals B	$\kappa(t_{A_1} - t_{B_1}) \cdot \kappa(t_{A_2} - t_{B_2})$	

replacement of those used in the definitions of temporal relations in Table 5. Step function $g(s)$ can be softened as, e.g., a triangular function or a logistic function, i.e.,

$$g(s) = \min(1, \max(0, \beta s + \tfrac{1}{2})),$$
$$\text{or} \quad g(s) = \frac{1}{1 + \exp(-\beta s)}. \tag{11}$$

Delta function $\kappa(s)$ can be softened as a triangular function or a Laplace density function, i.e.,

$$\kappa(s) = \max(0, \min(\tfrac{s}{\gamma^2} + \tfrac{1}{\gamma}, -\tfrac{s}{\gamma^2} + \tfrac{1}{\gamma})),$$
$$\text{or} \quad \kappa(s) = \frac{\exp(-|s|/\gamma)}{\gamma}. \tag{12}$$

Decaying parameters β and $\gamma \geq 1$ can be pre-specified or treated as unknown parameters, which can be learned from data by maximizing the likelihood. In this paper, we pre-specify these decaying parameters and make them frozen in the learning process (i.e., both the master and the subproblem).

C Proof of the Likelihood

Given a realization of all predicates $\{\mathcal{H}(t)\}_{0<t<T}$, one can write out the likelihood function in terms of the intensity function as follows.

For the head predicate Y, denote the dual conditional intensity function as $\lambda^*(t)$ and $\mu^*(t)$. Let $p(t_{n+1}|\mathcal{H}_{t_n}, y(t_n) = 0)$ and $p(t_{n+1}|\mathcal{H}_{t_n}, y(t_n) = 1)$ be the *conditional density function* of the next event time t_{n+1} given history and $y(t_n) = 0$, $y(t_n) = 1$, respectively. Let $F(t|\mathcal{H}_{t_n}, y(t_n) = 0)$ and $F(t|\mathcal{H}_{t_n}, y(t_n) = 1)$ be the corresponding cumulative distribution function for any $t > t_n$.

Based on the definition of the conditional transition intensity (or called hazard function), we have

$$\lambda^*(t) = \frac{p(t|\mathcal{H}_{t_n}, y(t_n) = 0)}{1 - F(t|\mathcal{H}_t, y(t_n) = 0)},$$
$$\text{and} \quad \mu^*(t) = \frac{p(t|\mathcal{H}_{t_n}, y(t_n) = 1)}{1 - F(t|\mathcal{H}_{t_n}, y(t_n) = 1).} \tag{13}$$

From (13), we have

$$\lambda^*(t) = -\frac{d}{dt} \log(1 - F(t|\mathcal{H}_t, y(t_n) = 0)),$$
$$\mu^*(t) = -\frac{d}{dt} \log(1 - F(t|\mathcal{H}_t, y(t_n) = 1)).$$

Integrating both sides, we can get the conditional density and the cumulative distribution function,

$$p(t|\mathcal{H}_{t_n}, y(t_n) = 0) = \lambda^*(t) \exp\left(-\int_{t_n}^{t} \lambda^*(s)ds\right),$$
$$F(t|\mathcal{H}_{t_n}, y(t_n) = 0) = 1 - \exp\left(-\int_{t_n}^{t} \lambda^*(s)ds\right),$$
$$p(t|\mathcal{H}_{t_n}, y(t_n) = 1) = \mu^*(t) \exp\left(-\int_{t_n}^{t} \mu^*(s)ds\right),$$
$$F(t|\mathcal{H}_{t_n}, y(t_n) = 1) = 1 - \exp\left(-\int_{t_n}^{t} \mu^*(s)ds\right).$$

Let $t_0 = 0$. Given the transition times $(t_1, t_2, \ldots, t_n)$, and suppose $y(t_0) = 0$, $y(t_n) = 1$ and the head predicate is still in state 1 at time T, the likelihood function can be factorized into all the conditional densities of each points given all points before it, i.e., the likelihood function is
$\mathcal{L} = p(t_1|\mathcal{H}_{t_0}, y(t_0) = 0)p(t_2|\mathcal{H}_{t_1}, y(t_1) = 1) \cdots p(t_n|\mathcal{H}_{t_{n-1}}, y(t_{n-1}) = 0)(1 - F(t|\mathcal{H}_{t_n}, y(t_n) = 1))$.
Plugging in the conditional density function and the cumulative distribution function, the likelihood is expressed as,

$$\begin{aligned}\mathcal{L} \quad &= \lambda^*(t_1) \exp\left(-\textstyle\int_0^{t_1} \lambda^*(s)ds\right) \cdot \mu^*(t_2) \exp\left(-\textstyle\int_{t_1}^{t_2} \mu^*(s)ds\right) \\ &\quad \cdots \lambda^*(t_n) \exp\left(-\textstyle\int_{t_{n-1}}^{t_n} \lambda^*(s)ds\right) \cdot \exp\left(-\textstyle\int_{t_n}^{t} \mu^*(s)ds\right),\end{aligned}$$

which completes the proof.

D Optimality Condition and Complementary Slackness

We will provide more descriptions on the optimality condition and the complementary slackness, which provides a sound guarantee to our learning algorithm.

Given the original restricted convex problem,

$$\text{P}_{\text{Master}}: \quad \boldsymbol{w}^*, b_0^*, b_1^* = \operatorname*{argmin}_{\boldsymbol{w}, b_0, b_1} -\ell(\boldsymbol{w}, b_0, b_1) + \sum\nolimits_{f \in \bar{\mathcal{F}}} c_f w_f; \quad s.t. \quad w_f \geq 0, \quad f \in \bar{\mathcal{F}} \tag{14}$$

where parameter c_f depends on the complexity of rule f, such as the number of predicates involved in f (i.e., rule length).

The Lagrangian of the original master problem is

$$L(\boldsymbol{w}, b_0, b_1, \boldsymbol{\nu}) = -\ell\left(\boldsymbol{w}, b_0, b_1\right) + \sum_{f \in \bar{\mathcal{F}}} c_f w_f - \sum_{f \in \bar{\mathcal{F}}} \nu_f w_f, \tag{15}$$

where $\nu_f \geq 0$ is the Lagrange multiplier associated with the non-negativity constraints of w_f. As it is a convex problem and strong duality holds under mild conditions. Define $\boldsymbol{w}^*, b_0^*, b_1^*$ as the primal optimal, and $\boldsymbol{\nu}^*$ as the dual optimal, then:

$$\begin{aligned}-\ell(\boldsymbol{w}^*, b_0^*, b_1^*) &= \inf_{\boldsymbol{w}, b_0, b_1} L(\boldsymbol{w}, b_0, b_1, \boldsymbol{\nu}^*) \qquad \text{(strong duality)} \\ &= \inf_{\boldsymbol{w}, b_0, b_1} \left(-\ell\left(\boldsymbol{w}, b_0, b_1\right) + \sum_{f \in \bar{\mathcal{F}}} c_f w_f - \sum_{f \in \bar{\mathcal{F}}} \nu_f^* w_f \right) \\ &\leq -\ell\left(\boldsymbol{w}^*, b_0^*, b_1^*\right) + \sum_{f \in \bar{\mathcal{F}}} \lambda_f w_f^* - \sum_{f \in \bar{\mathcal{F}}} \nu_f^* w_f^* \\ &\leq -\ell\left(\boldsymbol{w}^*, b_0^*, b_1^*\right) + \sum_{f \in \bar{\mathcal{F}}} \lambda_f w_f^*.\end{aligned} \tag{16}$$

Therefore, $\sum_{f \in \bar{\mathcal{F}}} \nu_f^* w_f^* = 0$, for $f \in \bar{\mathcal{F}}$. This implies the *complementary slackness*, i.e.,

$$w_f^* = 0 \Rightarrow \nu_f^* \geq 0, \qquad w_f^* > 0 \Rightarrow \nu_f^* = 0 \tag{17}$$

Given the Karush-Kuhn-Tucker (KKT) conditions, gradient of Lagrangian $L(\boldsymbol{w}^*, b_0^*, b_1^*, \boldsymbol{\nu}^*)$ w.r.t. $\boldsymbol{w}^*, b_0^*, b_1^*$ vanishes, i.e.,

$$\nu_f^* := -\left.\frac{\partial \ell(\boldsymbol{w}, b_0, b_1)}{\partial w_f}\right|_{\boldsymbol{w}^*, b_0^*, b_1^*} + c_f. \tag{18}$$

In summary, combining conditions (17) and (18), we obtain the optimalitiy condition,

1. if $w_f^* > 0$, then $\nu_f^* = 0$;
2. if $w_f^* = 0$, then $\nu_f^* \geq 0$,

where the gradient ν_f^* can be computed via (18).

E ALGORITHM BOX

Our TELLER alternates between solving a restricted master problem and a subproblem. We summarize the algorithm in Algorithm 1 and Algorithm 2. RMP indicates the Restricted Master Problem used to update model parameters. SP refers to the Sub-Problem used to construct a new rule. Here we use RAFS as our search scheme.

Algorithm 1: TELLER (RAFS)

Input: TimeLimit, MaxRuleLen
Output: ruleSet

```
stack ← empty stack;
ruleSet ← empty set;
b ← 0;
w ← 0;
b, w ← RMP(b, w, ruleSet); // Initialize weights and bases.
while RunTime≤ TimeLimit do
    if stack.isEmpty() then
        NewRule ← SP(b, w, ruleSet, None); // Search simple rule with length
            = 1.
        if NewRule is None then
            break; // If simple rule does not exist, algorithm ends.
    else
        RuleToExtend ← stack.top();
        NewRule ← SP(b, w, ruleSet, RuleToExtend);// Try to extend this rule.
        if NewRule is None then
            stack ← empty stack;// If this rule can not be extended, do
                not revisit it.
        if len(NewRule)=MaxRuleLen then
            stack ← empty stack;// If this rule reaches the maximum rule
                length, stop extending it.
    if NewRule then
        ruleSet.add(NewRule);
        stack.push(NewRule);
        b, w ← RMP(b, w, ruleSet);// After adding new rule, update
            weights and bases
return ruleSet
```

Algorithm 2: SubProblem (SP)

Input: $\boldsymbol{b}, \boldsymbol{w}$, ruleSet, RuleToExtend
Output: NewRule

TempRelSet $\leftarrow \{R_{\text{be}}, R_{\text{eq}}, R_{\text{me}}, \ldots, \text{Null}\}$; `// Thirteen types of temporal relation and Null (i.e., no temporal relation constraint).`
bodyPredSet, Y $\leftarrow$ defined by dataset;
candRuleSet $\leftarrow$ empty set;
if RuleToExtend is None **then**
    `// Search simple rule with length = 1.`
    for $\text{sign}(Y)$ in $\{+, -\}$ **do**
        for X in bodyPredSet and $\text{sign}(X)$ in $\{+, -\}$ **do**
            for $R_{X,Y}$ in TempRelSet **do**
                candRuleSet.add($(\neg)Y \leftarrow (\neg)X \wedge R_{X,Y}$);
else
    `// Try to extend input rule.`
    for X in bodyPredSet.difference(RuleToExtend) and for $\text{sign}(X)$ in $\{+, -\}$ **do**
        for X' in RuleToExtend **do**
            for $R_{X,X'}$ in TempRelSet **do**
                candRuleSet.add($(\neg)Y \leftarrow$ RuleToExtend $\wedge$ $(\neg)X$ $(\wedge_{X' \text{ in RuleToExtend}} R_{X,X'})$);
optValue, optRule $\leftarrow$ Evaluate the subproblem objective and choose the smallest one from the candRuleSet;
if optValue < 0 **then**
    `// This rule is a valid rule, i.e.  it can improve RMP.`
    NewRule $\leftarrow$ optRule;
else
    NewRule $\leftarrow$ None;
return NewRule

F SEARCH SCHEME FOR SUBPROBLEM: RULE-ADDITION-FIRST SEARCH

The search scheme can also be the Rule-Addition-First Search (RAFS). For the RAFS, the algorithm starts with an empty rule set and first focuses on discovering all rules with only one body predicate. Then assign each of them a score that equals the objective function of the subproblem (pricing problem) applied to the rule. To expand the rules from length l to $l + 1$, we do the following: we process all generated rules that have l predicates in increasing order of their score (since we aim to find a negative reduced cost), and for each such rule, we create new rules by appending an additional predicate together with the associated temporal relations. Whenever we find a rule with a negative reduced cost, we add it to the current rule list. When our enumeration terminates, we return the best rules generated by the heuristic before proceeding to the next value of l. When none of the rules with length l can be extended to $l + 1$, we proceed to expand rule from length $l + 1$ to $l + 2$. We may also pre-specify a maximum rule length and set a time limit to the algorithm.

G SYNTHETIC EXPERIMENT RESULTS

Dataset description: For synthetic experiment, we systematically considered 12 settings, where settings-$\{1, 7, 9, 10, 11, 12\}$ are reported as dataset-$\{1, 2, 3, 4, 5, 6\}$ in main text Section 4.2. Each setting corresponds to different rule weights, rule length and number, type of temporal relation, and intensity of free predicates.

To verify the similarity between the ground truth rules and the generated rules, we further utilize the Jaccard coefficient to measure the degree of consistency between the generated rules and the truth rules. For the i-th ground truth rule data, let $\hat{U}_i$ be the set of rules in the i-th ground truth rule and U_i be the i-th rules generated by our method. The Jaccard is defined as $\frac{|U_i \cap \hat{U}_i|}{|U_i \cup \hat{U}_i|}$.

The Jaccard similarities (range from 0 to 1) are reported in Table 6 and Fig. 7. We explore how the sample size (600, 1200, 2400) and the search method (REFS and RAFS) will impact the performance. We plot the Jaccard of our method in different sample sizes and search methods on 12 settings (see Table 7 for descriptions). The results are summarized in Fig. 7. We find that the performance gradually improves with the increase of sample size, which verifies that sufficient data can benefit the learning performance.

As mentioned in Section 3.3, REFS will always extend the longest existing rule, and RAFS is always extending the shortest existing rule. As shown in Fig. 7, REFS achieves similar performance with RAFS on most settings.

Table 6: Synthetic Data: Jaccard similarity with the Ground Truth Rules.

Setting	REFS-600	REFS-1200	REFS-2400	RAFS-600	RAFS-1200	RAFS-2400
Setting-1	0.222	0.972	1.000	0.200	1.000	0.500
Setting-2	0.059	0.000	0.250	0.200	0.200	0.222
Setting-3	0.182	0.286	0.750	0.286	0.375	0.750
Setting-4	0.111	0.429	0.600	0.117	0.428	0.600
Setting-5	0.200	0.571	1.000	0.125	0.500	1.000
Setting-6	0.250	0.667	0.750	0.222	0.333	0.600
Setting-7	0.250	0.750	0.750	0.090	0.750	0.600
Setting-8	0.167	0.000	0.286	0.071	0.00	0.200
Setting-9	0.750	0.600	1.000	0.600	0.750	1.000
Setting-10	0.059	0.231	0.600	0.111	0.154	0.750
Setting-11	0.444	0.571	0.800	0.364	0.444	0.800
Setting-12	0.375	0.429	0.750	0.091	0.429	1.000

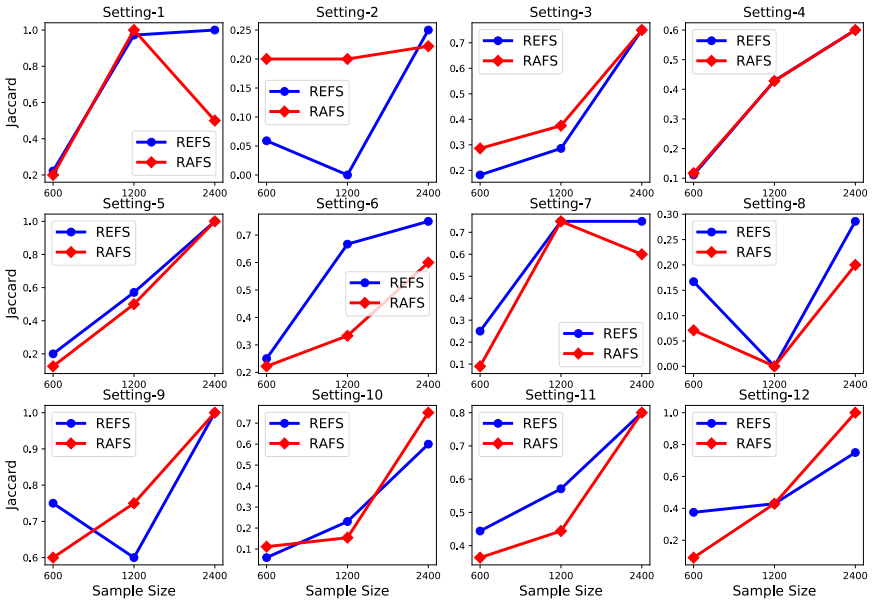


Figure 7: Jaccard similarity on Synthetic Data: with REFS Results in Blue and RAFS Results in Red.

Table 7: Synthetic Data Setting and Observations.

Dataset	Observation
Setting-1: Same body predicate intensity $\boldsymbol{w} = [1, 1, 1]$ $\boldsymbol{\lambda} = [1, 1, 1, 1]$	Hard scenario since all body predicates have the same occurrence rate. TELLER can still recover the rules (Table 2) and their weights.
Setting-2: Low important weights $\boldsymbol{w} = [0.5, 0.5, 0.5]$ $\boldsymbol{\lambda} = [1, 1, 1, 1]$	Lowering the important weights enhances the challenges in learning.
Setting-3: High important weights $\boldsymbol{w} = [1.5, 1.5, 1.5]$ $\boldsymbol{\lambda} = [1, 1, 1, 1]$	For relatively high rule weights, TELLER can accurately recover the rules and their weights.
Setting-4: Add rule length $\boldsymbol{w} = [1, 1, 2]$ $\boldsymbol{\lambda} = [1, 1, 1, 1]$	Adding rule length enhances the challenges in learning.
Setting-5: Add one rule $\boldsymbol{w} = [1, 1, 1, 1]$ $\boldsymbol{\lambda} = [1, 1, 1, 1]$	Adding one rule does not harm the rule recovery and learning performance.
Setting-6: Add one dummy predicate $\boldsymbol{w} = [1, 1, 1, 1]$ $\boldsymbol{\lambda} = [1, 1, 1, 1, 0.2]$	Adding dummy predicates enhances the challenges in learning.
Setting-7: Different body predicate intensities $\boldsymbol{w} = [1, 1, 1]$ $\boldsymbol{\lambda} = [0.6, 0.8, 1.2, 1.4]$	Body predicates have different occurrence rates. The MAE results outperform **Setting-1**
Setting-8: Different intensities and low weights $\boldsymbol{w} = [0.5, 0.5, 0.5]$ $\boldsymbol{\lambda} = [0.6, 0.8, 1.2, 1.4]$	Lowering the important weights enhances the challenges in learning, but different intensities lower the challenges.
Setting-9: Different intensities and high weights $\boldsymbol{w} = [1.5, 1.5, 1.5]$ $\boldsymbol{\lambda} = [0.6, 0.8, 1.2, 1.4]$	Both high weights and different intensities lower the challenges.
Setting-10: Different intensities and add rule length $\boldsymbol{w} = [1, 1, 2]$ $\boldsymbol{\lambda} = [0.6, 0.8, 1.2, 1.4]$	Both adding rule length and different intensities enhance the challenges.
Setting-11: Different intensities and add one rule $\boldsymbol{w} = [1, 1, 1, 1]$ $\boldsymbol{\lambda} = [0.6, 0.8, 1.2, 1.4]$	Both adding one rule and different intensities lower the challenges.
Setting-12: Different intensities and add one dummy predicate $\boldsymbol{w} = [1, 1, 1]$ $\boldsymbol{\lambda} = [0.6, 0.8, 1.2, 1.4, 0.2]$	Adding dummy predicate enhances the challenges in learning, but different intensities lower the challenges.

H REAL EXPERIMENT: MIMIC

H.1 SUPPLEMENTAL INFORMATION

MIMIC-III is a dataset released under PhysioNet Credentialed Health Data License 1.5.0[1]. It was approved by the Institutional Review Boards of Beth Israel Deaconess Medical Center (Boston, MA) and the Massachusetts Institute of Technology (Cambridge, MA). The requirement for individual patient consent was waived because all the patient health information was deidentified. We manually checked that this data do not contain personally identifiable information or offensive content.

H.2 HYPERPARAMETERS AND EXPERIMENT ENVIRONMENT

For the MIMIC dataset, we limit the maximum rule length to be 3, and the maximum #rules to be 20. The learning rate in solving the restricted master problem is $\times 10^{-4}$. The master problem is optimized by the SGD type of algorithm and we choose to use the projected gradient descent to take care of the weight constraints. The batch size is 64. Each time we solve the subproblem, we randomly selected 50% of the training data (i.e., patient sequences) to evaluate the subproblem objective. To exclude noisy and irrelevant rules, we clip the learned weights and discard these rules with weights smaller than is 10^{-2} in solving the restricted master problem. To further reduce the number of candidate rules, we set a threshold to the subproblem gain as 5×10^{-3}, i.e., we only include the candidate rules with negative cost smaller than -5×10^{-3}. Our model is trained and evaluated using 16 processes in parallel, on a server with a Xeon W-3175X CPU.

For the neural-based baselines, we set: 1) TR-GRU with 4 hidden states and 4009 trainable weights, 2) RPPN with 64 hidden states, 64 embedding dimensions, and 33277 trainable weights, 3) CAUSE with 16 hidden states and 19312 trainable weights, 4) NHP with 8 hidden states, 814 trainable weights,

[1] https://physionet.org/content/mimiciii/view-license/1.4/

5)THP with 8 hidden states, 3608 trainable weights, 6) RMTPP with 32 hidden states, 5440 trainable weights, and 7) IPP with 3 Gaussian kernels, $\mathbf{c} = [1, 1, 1], \sigma = [1, 0.8, 0.5]$.

H.3 PREDICATE DEFINITION IN MIMIC

Table 8: Defined Predicates in Our MIMIC-III Experiment.

Lab Measurements	Low/Normal/High-SysBP Low/Normal/High-SpO2SaO2 Low/Normal/High-CVP Low/Normal/High-SVR Low/Normal/High-Potassium Low/Normal/High-Sodium Low/Normal/High-Chloride Low/Normal/High-BUN Low/Normal/High-Creatinine Low/Normal/High-CRP Low/Normal/High-RBCcount Low/Normal/High-WBCcount Low/Normal/High-ArterialpH Low/Normal/High-ArterialBE Low/Normal/High-ArterialLactete Low/Normal/High-HCO3 Low/Normal/High-SvO2ScvO2
Output	LowUrine
Input	Colloid, Crystalloid, Water
Drugs	Norepinephrine, Epinephrine, Dobutamine, Dopamine, Phenylephrine
Suvival Condition	Survival
Temporal Relation Type	Before, Equal

H.4 DISCOVERED TEMPORAL LOGIC RULES

The discovered rules to explain the real-time urine have been reported in Section 4 Table 4. We list the discovered rules relate to the survival condition here in Table 9.

Table 9: Learned Rules with `Survival` head predicate for Sepsis Patients in MIMIC-III

Weight	**Rule**
0.95	**Rule 1:** NotSurvival ← NormalSVR ∧ Epinephrine ∧(Epinephrine Before NotSurvival) ∧(NormalSVR Before NotSurvival)
0.91	**Rule 2:** NotSurvival ← HighArterialBe ∧(HighArterialBe Before NotSurvival)
0.82	**Rule 3:** NotSurvival ← ∧HighBUN ∧ Phenylephrine ∧(Phenylephrine Before NotSurvival) ∧(HighBUN Before NotSurvival)
0.51	**Rule 4:** NotSurvival ← HighSodium ∧(HighSodium Before NotSurvival)
0.53	**Rule 5:** NotSurvival ← HighSodium ∧ Norepinephrine ∧(HighSodium Before NotSurvival) ∧(Norepinephrine Before NotSurvival)
0.89	**Rule 6:** Survival ← NormalAterialPH ∧(NormalAterialPH Before Survival)
0.55	**Rule 7:** NotSurvival ← HighPotassium ∧(HighPotassium Before NotSurvival)
1.19	**Rule 8:** NotSurvival ← ∧HighPotassium ∧ Colloid ∧(HighPotassium Before NotSurvival) ∧(Colloid Before NotSurvival)
1.00	**Rule 9:** NotSurvival ← HighlAterialPH ∧ UseNorepinephrine ∧(HighlAterialPH Before NotSurvival) ∧(HighlAterialPH Before NotSurvival)
0.61	**Rule 10:** NotSurvival ← HighHCO3 ∧ (HighHCO3 Before Survival)

I Real Experiment: Crime

Crime Incident Reports are provided by Boston Police Department to document the type of each incident as well as when and where it occurred (of Innovation & Technology, 2015).

I.1 Supplemental Information

This dataset is released by the Boston Department of Innovation and Technology under Open Data Commons Public Domain Dedication and License (PDDL). The publisher does not discuss how the data was collected and whether consent was obtained. We manually checked that this data does not contain personally identifiable information or offensive content.

Predicate Definition and Dataset statistics. We are interested in the top four most frequent crime types: vandalism, theft from motor vehicles, assault, and shoplifting. Another ten predicates are defined to describe the occurrence time properties, such as whether it is in the morning or the afternoon, on weekdays or weekends, in spring or winter, etc. The set of defined predicates is displayed in Table 10. We consider all the crime reports from June 2015 to May 2021 and split the data into 1879 sequences according to days. We randomly choose 80% of these sequences as training data and the remaining as testing data. On average, each sequence contains 46.03 events.

Table 10: Defined Predicates for Crime.

Period of Crime	`Spring`, `Summer`, `Autumn`, `Winter`, `Weekday`, `Weekend`, `Morning`, `Afternoon`, `Evening`, `Night`
Crime Types	`Vandalism`, `TheftFromMV`, `Assault`, `Shoplifting`
Temporal Relation Type	`Before`, `Equal`

I.2 Hyperparameters and Experiment Environment.

Our model is trained and evaluated using 16 processes in parallel, on a server with a Xeon W-3175X CPU. For this Crime dataset, we limit the maximum rule length to be 2, and the maximum #rules to be 20. The learning rate used in updating model parameters in the restricted master problem is 10^{-4}. The master problem is optimized by SGD with a batch size of 64. The subproblem objective function

is evaluated on the entire training data. To exclude noisy and irrelevant rules, we clip the learned weights and discard these rules with weights smaller than is 10^{-2} in solving the restricted master problem. To further reduce the number of candidate rules, we set a threshold to the subproblem gain as $\times 10^{-2}$, i.e., we only include the candidate rules with negative cost smaller than $-\times 10^{-2}$.

For the non-parametric baselines, we set: 1) TR-GRU with 4 hidden states and 1129 trainable weights, 2) RPPN with 64 hidden states, 64 embedding dimensions, and 27037 trainable weights, 3) CAUSE with 16 hidden states and 5872 trainable weights, 4) NHP with 8 hidden states, 382 trainable weights, 5)THP with 8 hidden states, 2408 trainable weights, 6)RMTPP with 32 hidden states, 2240 trainable weights, and 7)IPP with 3 Gaussian kernels, $\mathbf{c} = [1, 1, 1], \sigma = [1, 0.8, 0.5]$.

I.3 Prediction Accuracy

Table 11: Crime: MAE of Event Time Prediction.

Method	Vandalism	Larceny TheftFromMV	Assault	Larceny Shoplifting
RPPN	0.881	1.137	1.185	0.777
HExp	0.761	0.949	1.912	**0.704**
TR-GRU	0.759	1.351	1.400	1.092
CAUSE	0.962	1.127	1.206	0.892
NHP	**0.613**	1.300	1.887	1.269
THP	0.973	1.043	**0.957**	0.939
RMTPP	0.874	1.021	1.059	0.763
IPP	0.908	1.274	1.508	1.179
TELLER	0.770	**0.826**	1.465	0.710

The mean absolute error (MAE) of the predicted event times are displayed in Table 11. Our model outperforms other models in predicting TheftFromMV. It has a comparable performance with the Hawkes baseline on the remaining three tasks. We can think of Hawkes process as a special case of our model and this is especially true if we restrict our model to learn short rules.

I.4 Discovered Rules

We displayed the discovered important rules in Table 12, from which one can observe the following crime patterns. Shoplifting and TheftFromMV may trigger Vandalism. One explanatory reason is that larceny may involve break-in and destruction of security devices (Rule 1, 2, and 3). Shoplifting triggers TheftFromMV. These two events are special types of larceny, and thus they may exhibit similar crime patterns (Rule 4). TheftFromMV exhibits self-exciting (i.e., clustering) patterns in summer. This is may due to that winter is extremely cold in Boston, and summer, however, is a nice period of the year for outdoor activities and thus motor vehicles may pour into the area, which potentially increases the likelihood of theft (Rule 5). Assault shows a self-exciting pattern on weekends. People tend to have more social activities on weekends, which increases the possibility of domestic violence and affray (Rule 6). Shoplifting triggers Assault. This might be explained by the fact that if shoplifting is spotted at the scene, the thief may have a physical conflict with the security guard (Rule 7). TheftFromMV and Vandalism will work together to trigger Shoplifting, which is consistent with our previous observations 1 and 2 (Rule 8 and 9). Shoplifting is self-exiting on weekdays. Stores are busier on weekends than on weekdays, and thus weekdays might be a better choice for shoplifters (Rule 10).

Table 12: Temporal Logic Rules Discovered for Four Types of Crime Events.

Weight	**Rule**
3.84	**Rule 1:** Vandalism ← Shoplifting ∧(Shoplifting Before Vandalism)
1.83	**Rule 2:** Vandalism ← TheftFromMV ∧(TheftFromMV Before Vandalism)
0.43	**Rule 3:** Vandalism ← TheftFromMV ∧ Shoplifting ∧(Shoplifting Before Vandalism) ∧(TheftFromMV Before Vandalism)
2.46	**Rule 4:** TheftFromMV ← Shoplifting∧ (Shoplifting Before TheftFromMV)
1.42	**Rule 5:** TheftFromMV ← TheftFromMV ∧ Summer ∧(TheftFromMV Equal Summer) ∧(TheftFromMV Before TheftFromMV)
0.40	**Rule 6:** Assault ← Assault ∧ Weekend ∧(Weekend Equal Assault) ∧(Assault Equal Assault)
3.62	**Rule 7:** Assault ← Shoplifting ∧ Assault ∧(Shoplifting Before Assault) ∧(Assault Before Assault)
1.67	**Rule 8:** Shoplifting ← TheftFromMV ∧(TheftFromMV Before Shoplifting)
1.14	**Rule 9:** Shoplifting ← Shoplifting ∧ Vandalism ∧(Vandalism Before Shoplifting) ∧(Shoplifting Before Shoplifting)
1.30	**Rule 10:** Shoplifting ← Shoplifting ∧ Weekday ∧(Shoplifting Before Shoplifting) ∧(Weekday Equal Shoplifting)