Linear Temporal Public Announcement Logic: A New Perspective for Reasoning About the Knowledge of Multi-classifiers

Amirhoshang Hoseinpour Dehkordi\textsuperscript{1} · Majid Alizadeh\textsuperscript{2} · Ali Movaghar\textsuperscript{3}

Received: 14 October 2022 / Revised: 25 January 2023 / Accepted: 26 January 2023 / Published online: 7 March 2023
© The Author(s) under exclusive licence to Iranian Mathematical Society 2023

Abstract
In this paper, a formal transition system model is presented called Linear Temporal Public Announcement Logic (LTPAL) to extract knowledge in a classification process. The model combines Public Announcement Logic (PAL) and Linear Temporal Logic (LTL). For this purpose, first, an epistemic logic model is created to capture information gathered by classifiers in single-framed data input. Next, using LTL, classifiers are considered for data stream inputs. Then, a verification method is proposed for such data streams. Finally, we formalize natural language properties in LTPAL with a video-stream object detection sample.

Keywords Temporal logic · Epistemic logic · Public announcement logic · Verification · Classifier

Mathematics Subject Classification 03B42 · 03B44 · 03B45 · 03B65 · 68T27 · 68T42 · 03B70 · 68T20

Communicated by Saeid Maghsoudi.

\textsuperscript{\textcopyright} Majid Alizadeh
majidalizadeh@ut.ac.ir

Amirhoshang Hoseinpour Dehkordi
amir.hoseinpour@ipm.ir

Ali Movaghar
movaghar@sharif.edu

\textsuperscript{1} School of Computer Science, Institute for Research in Fundamental Sciences, 19538-33511 Tehran, Tehran, Iran

\textsuperscript{2} School of Mathematics, Statistics and Computer Science, College of Science, University of Tehran, 14155-6455 Tehran, Tehran, Iran

\textsuperscript{3} Department of Computer Engineering, Sharif University of Technology, 11155-9517 Tehran, Tehran, Iran
1 Introduction

Nowadays, classification is inseparable from real-world applications. The presence of errors increases the necessity for better interpretation and comprehension of such algorithms. Due to the probabilistic nature of most of these approaches, the reasoning about results remains complicated.

Reasoning is one of the most studied and accepted dimensions of AI [10, 17, 19, 22]. Furthermore, it inspires researchers to target visual data understanding, question answering, epidemic broadcasting protocols, and natural language queries [1, 8, 11]. For instance, a question-answering (QA) reasoning system was developed by Bauer et al., who introduced an algorithm to select the best paths of common sense knowledge and get the whole inference required for QA [2]. The method works fine for specific applications. However, it provides no general solution for QA problems. Based on the systematic analysis of popular knowledge resources and the knowledge-integration methods, Ma et al. developed a modeling solution [16]. The method, called “non-extractive commonsense QA”, employs ConceptNet (Speer et al. [21]) and the most recently introduced ATOMIC (Sap et al. [20]). More presently, based on human intelligence, the “CoLlision Events for Video REpresentation and Reasoning” method (CLEVRER) was developed by Yi and coworkers [24], which is a reasoning system for video streams. In this method, unlike former studies, “causal structures” are also taken into account. Moreover, this model can answer four main varieties of questions: descriptive (e.g., “what color?”), explanatory (e.g., “what is responsible for?”), predictive (e.g., “what will happen next?”), and counterfactual (e.g., “what if?”). Here, the solution for extracting features from the video frames is ResNet-50, a well-known classifier [12].

Thanks to the characteristics of the reasoning systems and the power of temporal logic in formalizing natural language, the model developed by Fong et al. deals with reasoning systems [7]. From another perspective, metric temporal logic (MTL), an extension of linear temporal logic (LTL), was examined to handle the stochastic state information [4]. Concerning typical AI-based problems, the employment of modal logic would be attractive. Recently, Dehkordi et al. developed an epistemic-based model for the extraction and aggregation of knowledge from classifiers [5]. Besides, the developed method can be optimized by load-balancing systems [6]. The model works perfectly for single-framed data. However, it addresses no data stream inputs. Dehkordi et al. [5] developed Algorithms 1 and 2 for knowledge-sharing modeling in a multi-classifier scenario. In a network of trusted classifiers (which share their knowledge correctly), the process of knowledge extraction from one input for an agent is developed in Algorithm 1. In other words, all output classes of the input, presented in the neighborhood set of \( \eta(x) \) (here referred to as \( \eta(x) \)), were collected. \( \eta(x) \) is a set of inputs in \( \epsilon \) vicinity (based on distance function \( d(x_0, x) \)). Here, the classifier function \( \mathcal{N}(x) \) yields the respecting output classes by \( \eta(x) \) input. Its knowledge contains the set of output classes of \( \mathcal{N}(x) \), which is represented by \( \mathcal{K} \) (as the knowledge set of the classifier). The output is robust if the size of the set of output classes of \( \mathcal{N}(x) \) is one. Therefore, from the agent’s perspective, the output concludes the robustness of \( x \). An agent’s perspective is knowledge, which is accessible by that agent. Let \( N \) be
the time complexity of each classifier; the time complexity of Algorithm 1 would be $O(N \times |\eta(x_0) \cup \mu(x_0)| \times |W|)$, where “$|A|$” is the cardinality of the set $A$.

**Algorithm 1** The Classifier Knowledge Calculator (CKC) function will calculate the knowledge produced by a classifier for an input point, using the neighborhood function and manipulation function

Let $N(x) = c$ be the function of the considered classifier and $c$ be the result class

1: function CKC($N$, $x_0$, $\eta$, $\mu$)
2: $\triangleright N, x_0, \eta, \mu$ are a classifier, an input point, neighborhood function, and manipulation function, respectively
3: $K \leftarrow \emptyset$
4: for all $x \in \eta(x_0) \cup \mu(x_0)$ do
5: $c \leftarrow N(x)$ $\triangleright c$ represents the respective possible world
6: if $c \notin K$ then
7: Add $c$ to $K$ set
8: end if
9: end for
10: if $|K| = 1$ then
11: return $1, K$
12: end if
13: return $0, K$
14: end function

In Algorithm 2, all captured knowledge is aggregated, and the reasoning is applied. After finding the intersection, if it contains just one class, the input is verified. Here, the MAS’s knowledge is inconsistent if the output is an empty set. In the case of more than one output class, agents are not capable of concluding the answer. However, they know the answer is among these classes. The time complexity of $K_S \leftarrow K_S \cap K$ is $O(|W|)$; thus, Algorithm 2 would run in $O(N \times |\eta(x_0) \cup \mu(x_0)| \times |W| \times |\mathcal{N}_S|)$. This algorithm models the single-framed data knowledge capturing in a multi-agent scenario.

This study aims to extend the expressiveness of the model developed by Dehkordi et al. [5] to present data streams. Moreover, a flexible reasoning system is suggested to render the knowledge achieved by classifiers. Here, the model is applied to the existing classifiers to translate the information attained from the results. Furthermore, the ability of the model to handle multi-knowledge flow in a multi-classifier scenario is assessed. For this purpose, various collaboration scenarios of classifiers (single agents or a group of agents) are studied. For single-framed data, our method verifies the properties (formulas) using the definition of “verification” mentioned in Dehkordi et al. [5]. Certain questions (in human language) are translated to the developed temporal formula (as properties) to check the adaptation and applicability of the proposed model. By checking the satisfiability of the temporal formula, the “verification of data streams” is presented.

The main contributions of this paper to the development of a reasoning model on top of a pre-existing intelligent system are summarized as follows:

1. We collect all the classifier’s provided answers by Algorithm 1.
Algorithm 2 The MAS Knowledge Sharing (MASKS) function will reason about the knowledge produced by classifiers

1: \textbf{function} \text{MASKS}(N_S, x_0, \eta, \mu) \\
2: \quad \triangleright \text{\(N_S, x_0, \eta, \mu\) are a group of classifiers, an input point, neighborhood function, and manipulation function, respectively} \\
3: \quad K_S \leftarrow \text{set of all output classes} \\
4: \quad \textbf{for all} \: \mathcal{N} \in N_S \textbf{ do} \\
5: \quad \quad \text{is}_\mathcal{N}\_\text{Robust}, K \leftarrow \text{CKC}(\mathcal{N}, x_0, \eta, \mu) \\
6: \quad \quad K_S \leftarrow K_S \cap K \\
7: \quad \quad \text{if} \: K_S = \emptyset \text{ then} \\
8: \quad \quad \quad \text{return} \: 0, \emptyset \\
9: \quad \quad \textbf{end if} \\
10: \quad \textbf{end for} \\
11: \quad \text{if} \: |K_S| = 1 \text{ then} \\
12: \quad \quad \triangleright \text{Check whether the knowledge can verify the input or if more knowledge is needed} \\
13: \quad \quad \text{return} \: 1, K_S \\
14: \quad \textbf{end if} \\
15: \quad \text{return} \: 0, K_S \\
16: \textbf{end function}

2. Then, using the method of Dehkordi et al. [5], we construct a multi-agent epistemic logic Kripke model to find all possible outcomes of multi-classifiers, Algorithm 2. Here, we can investigate the verification for single-frame data.

3. By letting each data stream frame as a single-frame input, we can create a Kripke model for each frame. Then, a transition system can be created by placing the Kripke models in hierarchical order. Here, we develop a logic for the transition system by combining temporal logic (Gerth et al. [9]) with the epistemic logic model. It leads us to extract all possible knowledge sequences represented in data streams.

4. Finally, the situation of a formula (equivalently a property) is determined for data streams.

The structure of the paper is as follows. In Sect. 2, the formal verification of classifiers is defined. A PAL model is defined in Sect. 3 to capture the information from single-framed data. Finally, in Sect. 4, an extension of the model (called "Linear Temporal Public Announcement Logic", LTPAL) is defined by combining PAL with LTL. LTPAL is used to reason the captured knowledge from data streams by multi-agent systems. To create a more convenient approach for defining the verification’s property, a procedure is suggested to formalize natural language in LTPAL, which can be served as an application.

2 Formal Verification of Classifiers

Generally, formal verification is a mathematical process to ensure that a model fulfills some property function in an environment [3]. Thus, to define the verification of classifiers, we should ensure that the defined property function holds for the classifier. The concept of creating a set using an indicator function was defined by Yang et al. [23]. Here, we define the property set as follows:
Definition 2.1 A property set $\rho$ for elements of a set $X$ defines with an indicator function $f_\rho : X \to \{true, false\}$, in which $\rho(X) = \{x \in X \mid f_\rho(x) = true\}$.

A single-object detector classifier is a classifier detecting a single object from an input. In some critical cases, single-object classifiers provide a set of answers because the confidence level of the classification process does not satisfy the minimum requirements. For example, in the YOLO, the classifier threshold can be defined, and the outputs are a set of answers with scores over the threshold (Redmon et al. [18]). Another study created a set of inputs by discovering a neighborhood and manipulation set for the inputs. In this study, the outcome was all the outputs of the classifier for the neighborhood and manipulation set (Huang et al. [14]). In this study, the property set is denoted by $\rho(X)$, and the set of answers provided by classifier $G$ for input $x$ considering the property $\rho$ is represented by the notation $[\rho(x)]_G$. By definition, a single-object classifier should provide a single output class. The concept of the verification of property in classifiers was defined by Dehkordi et al. [5]. We explain these within the literal context of this paper in the definitions as follows:

Definition 2.2 A property $\rho(x)$ for a single-object detector classifier $G$ is called verified exactly when $| [\rho(x)]_G | = 1$.

Definition 2.3 A property $\rho(x)$ for a set of single-object detector classifiers $A_G = \{G_1, \ldots, G_n\}$ is called verified exactly when $| \bigcap_{G \in A_G} [\rho(x)]_G | = 1$.

In a multi-agent scenario, a property for single-object detector classifiers is verified when multiple classifiers agree about a single answer. For more detail about these prerequisites, see [5].

Note that we can assume a multi-object classifier as multiple single-object detector classifiers and the definitions are identical for each single-object detector classifier. Here, we use the term classifier instead of the single-object detector classifier for simplicity.

We define the verification for data streams by defining the properties in the form of LTPAL formulas in Sect. 4.

3 Public Announcement Logic

In this section, we introduce a logical model for interpreting single-framed data. The model is based on PAL’s extension to the epistemic logic [5]. Here, we call this extension as PAL. It is applied to extract knowledge from single-framed data (i.e., an image).

Let us first introduce the syntax and semantics of the logic.

Definition 3.1 (Language of PAL) Let $Ag = \{1, \ldots, n\}$ be a finite set of agents. The syntax of the language PAL is as follows, in BNF:

$$\phi ::= p \mid \neg \phi \mid (\phi \land \phi) \mid K_i \phi \mid D_A \phi \mid [\phi]_G \phi,$$

where $p$ is a propositional variable (atomic formula) with a pair of forms $(x, c)$, in which $x$ denotes the input data, $c$ represents the target class, and $A \subseteq Ag$ is a subset.
of agents. We also notice that $K_i \phi$, read as “i-th classifier, knows $\phi$” ($i \in \text{set of all classifiers}$). In other words, the i-th classifier is assured about the truth value of $\phi$. Then, $D_A \phi$ read as “$\phi$ is a distributed knowledge in group A of classifiers”. The formula $D_A \phi$ holds exactly when aggregation of knowledge of agents (equivalently $\bigcap_{i \in A} R_i$) in the group A satisfies $\phi$ (if $A = \{i\}$, then $K_i$ is equal to $D_A$). Finally, the formula $[\psi]\phi$ reads as “after a correct announcement of $\psi$, $\phi$ is”.

Suppose that $G = \{G_1, \ldots, G_n\}$ is a finite set of classifiers, $X$ is the set of input points, and $C$ is the set of all output classes. A PAL Kripke model is a tuple $M = (W, R_1, \ldots, R_n, V)$, where $W$ is a set of worlds (hereabouts, the set $W$ represents all possible output results of the input data),

$$W := \{c \in C \mid \exists x \in X, \exists G \in G, \text{ such that, } c \in [\rho(x)]_G \cup \{\bar{c}\}\}.$$  

$R_i \subseteq W \times W$ is an equivalent relation between worlds for each classifier $G_i$ in $G$.

$$R_i(c) := \begin{cases} 
\{c\} & c \not\in [\rho(x)]_{G_i} \\
\{c' \mid c' \in [\rho(x)]_{G_i}\} \cup \{\bar{c}\} & c \in [\rho(x)]_{G_i}. 
\end{cases}$$

$R_i(\bar{c})$ are defined as follows:

$$R_i(\bar{c}) := \{c' \mid c' \in [\rho(x)]_{G_i}\} \cup \{\bar{c}\}.$$  

We add $\bar{c}$ to the set of worlds to find out the “class appearing as an output of any agent”. Since the classification of distinct input points is unrelated, we can consider a fixed input point $x$ and create the model consequently. The final model will be the disjoint union of the models for each input. The intended meaning of the $cR_i c'$ relation is that the i-th classifier cannot epistemically distinguish $c$ and $c'$. Finally, $V : W \rightarrow 2^{\text{Prop}}$ is the evaluation function specifying that the knowledge is represented in any world, where Prop is the set of all atoms. $V(c)$ and $V(\bar{c})$ are defined as follows:

$$V(c) := \{(x, c) \mid \exists G \in G, \text{ such that, } c \in [\rho(x)]_G\},$$

$$V(\bar{c}) := \{(x, c) \mid \exists G \in G, \text{ such that, } c \in [\rho(x)]_G\}.$$  

We extend the evaluation function $V$ to all formulas as follows:

- $\mathcal{M}, c \models p$ iff $p \in V(c)$,
- $\mathcal{M}, c \models \neg \phi$ iff $\mathcal{M}, c \not\models \phi$,
- $\mathcal{M}, c \models \phi \land \psi$ iff $\mathcal{M}, c \models \phi$ and $\mathcal{M}, c \models \psi$,
- $\mathcal{M}, c \models K_i \phi$ iff $\forall v \in R_i(c), \mathcal{M}, v \models \phi$,
- $\mathcal{M}, c \models D_A \phi$ iff $\forall v \in R_A(c), \mathcal{M}, v \models \phi$, where $R_A := \bigcap_{i \in A} R_i$,
- $\mathcal{M}, c \models [\psi]\phi$ iff $\mathcal{M}, c \models \psi$ implies $\mathcal{M}^\psi, c \models \phi$,

where $\mathcal{M}^\psi := (W^\psi, R_1^\psi, \ldots, R_n^\psi, V^\psi)$ with

- $W^\psi := \{c \in W \mid \mathcal{M}, c \models \psi\}$,
- $R_j^\psi := R_j \cap (W^\psi \times W^\psi)$ for all $j \in \{1, \ldots, n\}$, and
- $V^\psi(c) := V(c)$ for all $w \in W^\psi$.  

Springer
For any given model \( \mathcal{M} \) and any given formula \( \psi \), Algorithm 3 simply calculates the model \( \mathcal{M}^{\psi} \).

Now, we can show that a point is verified for classifiers exactly when its interpretation is valid in \( \mathcal{M} \).

**Theorem 3.2** [5] Suppose that \( \mathcal{G} = \{ G \} \) is a classifier, \( x \) is an input and \( c \) is an output class of \( G \). Then:

\[ [\rho(x)]_G = \{ c \} \text{ exactly when } \mathcal{M}, \bar{c} \models Kp_c, \text{ where } (x, c) \text{ denoted by } p_c. \]

**Proof** Only-if part. Suppose \( \mathcal{M}, c \models Kp_c \), where \( p_c \) is interpreted by \((x, c)\). As there is only one classifier and \( c \) is an output class, we have \( c \in [\eta(x) \cup \mu(x)]_G \). Suppose \( c' \in [\eta(x) \cup \mu(x)]_G \). So \( c' \in R(c) \) and we have \( \mathcal{M}, c' \models p_c \) by hypothesis, which means that \((x, c') \in V(c)\), thus \( c = c' \).

If-part. Suppose \([\eta(x) \cup \mu(x)]_G = \{ c \}\). Because \( c \) is an output of the classifier, \( c \in W \), and as it is the only output of classifier \( G \) on input \( x \), we have \( R(c) = \{ c \} \). Thus, it suffices to show that \( \mathcal{M}, c \models p_c \). As \((x, c) \in V(c)\), the statement is proved. \( \square \)

**Theorem 3.3** [5] Suppose that \( \mathcal{G} = \{ G_1, \ldots, G_n \} \) is a multi-classifier system such that \( \bigcap_{G \in \mathcal{G}} [\rho(x)]_G \neq \emptyset \), \( x \) is an input and \( c \) is an output class for classifiers in \( \mathcal{G} \), then:

\[ \bigcap_{G \in \mathcal{G}} [\rho(x)]_G = \{ c \} \text{ exactly when } \mathcal{M}, \bar{c} \models D_Gp_c, \text{ where } (x, c) \text{ denoted by } p_c. \]

**Proof** Only-if part. Suppose \( \mathcal{M}, \bar{c} \models D_Gp_c \), then for all \( c' \) in \( R_DG(\bar{c}) \) we have \( \mathcal{M}, c' \models p_c \). On the other hand, by the assumption there is an element \( d \) in \( \bigcap_{G \in \mathcal{G}} [\eta(x) \cup \mu(x)]_G \). It is enough to show that \( d = c \). We have \( d \in \bigcap_{G \in \mathcal{G}} R_G(\bar{c}) \).

Since \( \mathcal{M}, d \models p_c \), then \( V(d) = \{(x, c)\} \) which implies that \( d = c \).

If-part. Suppose \( \bigcap_{G \in \mathcal{G}} [\eta(x) \cup \mu(x)]_G = \{ c \} \). Because \( c \) is an output of the classifiers, \( c \in W \), and \( c \in [\eta(x) \cup \mu(x)]_G \) for all \( G \in \mathcal{G} \). On the other hand as it is the only common output of all the classifiers, for any other output class, there exist a classifier \( G \) that does not have it as output. So, \( R_DG(c) = \{ c, \bar{c} \} \). Therefore, it suffices to show that \( \mathcal{M}, \bar{c} \models p_c \). As \((x, c) \in V(c) \cap V(\bar{c})\), the statement is proved. \( \square \)

**Algorithm 3** The Single-Framed Knowledge Extraction (SFKE) function will reduce the model using the knowledge produced by classifiers.

Let \( \mathcal{M} \) be a Kripke model and \( \varphi \) a PAL formula

1. **function** SFKE(\( \mathcal{M}, \varphi \))

2. \( \triangleright \ V \text{ could be driven from the model } \mathcal{M} \)

3. **for all** possible world \( c \in \mathcal{M} \) **do**

4. **if** not PALS(\( \varphi, \mathcal{M}, c \)) **then**

5. Remove \( c \) from the model \( \mathcal{M} \).

6. Remove all relations \( cR_jc' \) and \( c'R_jc \) from the model \( \mathcal{M} \).

7. **end if**

8. **end for**

9. **return** \( \mathcal{M} \)

10. **end function**

In the following theorem, Algorithm 4 was developed to investigate the correctness of PAL formulas. Let \( \text{Depth}_\varphi \) and \( \text{Denop}_\varphi \) be the number of non-path (\( \neg, \land \)) and
path operators ($K_i$, $D_A$, $[]$) in formula $\phi$; the time complexity of Algorithm 4 would be $O(|W|^{|DePith_\phi| \times DeNop_\phi|}$. Consequently, the time complexity of Algorithm 3 is $O(|W| \times |W|^{|DePith_\phi| \times DeNop_\phi|}$.

**Theorem 3.4** Let $\mathcal{M} = (W, R_1, \ldots, R_n, V)$ be a Kripke model, $c \in W$ and $\phi$ a PAL formula. Then, $\mathcal{M}, c \models \phi$ if and only if $\text{PALS}(\phi, \mathcal{M}, c)$.

**Proof** We only prove the if direction; the other direction is proved similarly. The proof is obtained by induction on the complexity of $\phi$. Suppose that $\mathcal{M}, c \models \phi$, and $\phi \in \text{PAL}$.

- If $\phi$ is an atomic formula, then the if condition in line 3 would be satisfied, so line 4 would be executed. Here, we will check whether $\phi$ is in $V(c)$ or not. By definition, we have $\mathcal{M}, c \models p$ if and only if $p \in V(c)$.
- If $\phi \equiv \neg \phi_1$, then, in this case, the if condition in line 7 would be satisfied, so line 8 would be executed. At this line, we return the negation of $\text{PALS}(\phi_1, \mathcal{M}, c)$. Then, by induction $\text{PALS}(\phi_1, \mathcal{M}, c)$ if $\mathcal{M}, c \models \phi_1$.
- If $\phi \equiv \phi_1 \land \phi_2$, then the if condition at line 10 would be satisfied, and line 11 would be executed. At this line we return the conjunction of $\text{PALS}(\phi_1, \mathcal{M}, c)$ and $\text{PALS}(\phi_2, \mathcal{M}, c)$. By induction of $\text{PALS}(\phi_1, \mathcal{M}, c)$, if $\mathcal{M}, c \models \phi_1$ and $\text{PALS}(\phi_2, \mathcal{M}, c)$, then $\mathcal{M}, c \models \phi_2$.
- If $\phi \equiv K_i \phi_1$, then the if condition in line 13 would be satisfied, and line 14 would be executed. In this line, for every $c' \in R_i(c)$, $\text{PALS}(\phi_1, \mathcal{M}, c')$ would be calculated; $\text{true}$ would be returned if and only if they all are satisfied. By induction, we have $\text{PALS}(\phi_1, \mathcal{M}, c')$ if $\mathcal{M}, c' \models \phi_1$. By definition, we have $\forall c' \in R_i(c)$, $\mathcal{M}, c' \models \phi_1$ if $\mathcal{M}, c \models K_i \phi_1$. Therefore, for all $c' \in R_i(c)$, $\text{PALS}(\phi_1, \mathcal{M}, c')$ if $\mathcal{M}, c \models \phi_1$. Consequently, $\text{PALS}(\phi, \mathcal{M}, c)$ if $\mathcal{M}, c \models \phi$.
- If $\phi \equiv D_A \phi_1$, then the if condition at line 16 would be satisfied, and lines 17 and 18 would be executed. In line 17, first, $R_A$ would be calculated the same as $R_{DA}$ in the definition. Then, for every $c' \in R_A(c)$, $\text{PALS}(\phi_1, \mathcal{M}, c')$ would be calculated; $\text{true}$ would be returned if and only if they all are satisfied. By induction, we have $\text{PALS}(\phi_1, \mathcal{M}, c')$ if $\mathcal{M}, c' \models \phi_1$. By definition, we have $\forall c' \in R_{DA}(c)$, $\mathcal{M}, c' \models \phi_1$ if $\mathcal{M}, c \models D_A \phi_1$. Therefore, for all $c' \in R_i(c)$, $\text{PALS}(\phi_1, \mathcal{M}, c')$ if $\mathcal{M}, c \models D_A \phi_1$. Consequently, $\text{PALS}(\phi, \mathcal{M}, c)$ if $\mathcal{M}, c \models \phi$.
- If $\phi \equiv [\phi_2] \phi_1$, then the if condition at line 20 would be satisfied, and lines 21–25 would be executed. In line 21, first, we check if $\text{PALS}(\phi_2, \mathcal{M}, c)$ is false. Based on the definition, $\mathcal{M}, c \models \phi_2$ implies that $\mathcal{M}^{\phi_2}, c \models \phi_1$ would be $\text{true}$. Because, by induction we had $\text{PALS}(\phi_2, \mathcal{M}, c)$ if $\mathcal{M}, c \models \phi_2$. Otherwise, we could calculate $\mathcal{M}^{\phi_2}$ with the function $\text{SFKE}(\mathcal{M}, \phi_2)$, Algorithm 3. SFKE returns a modification of the Kripke model $\mathcal{M}$, in which all worlds $c$, $\mathcal{M}, c \not\models \phi_2$ are removed. Therefore, $\text{SFKE}(\mathcal{M}, \phi_2) \equiv \mathcal{M}^{\phi_2}$. By induction, we have $\text{PALS}(\phi_2, \mathcal{M}, c)$ if $\mathcal{M}, c \models \phi_2$ and $\text{PALS}(\phi_1, \mathcal{M}^{\phi_2}, c)$ if $\mathcal{M}^{\phi_2}, c \models \phi_1$. Based on the definition, we have $\mathcal{M}, c \models \phi_2$ implies $\mathcal{M}^{\phi_2}, c \models \phi_1$ if $\mathcal{M}, c \models [\phi_2] \phi_1$. Consequently, $\text{PALS}(\phi, \mathcal{M}, c)$ if $\mathcal{M}, c \models \phi$. \qed
Algorithm 4 The PAL Satisfaction function (PALS) will investigate the satisfaction of PAL formulas

Let $\phi$ be the PAL formula, $\mathcal{M}$ the Kripke model, and $c$ a world

1: function PALS($\phi, \mathcal{M}, c$)  
2: \>	ext{// $R_i$ and $V$ could be driven from the model $\mathcal{M}$} 
3: if $\phi$ is an atomic formula then 
4: \>	ext{return } \phi \in V(c) 
5: \>	ext{// This will be true if } \phi \in V(c) \text{ and false otherwise} 
6: end if 
7: if $\phi$ is in form of $\neg \psi$ then 
8: \> return $\neg$PALS($\psi, \mathcal{M}, c$) 
9: end if 
10: if $\phi$ is in form of $\psi_1 \land \psi_2$ then 
11: \> return PALS($\psi_1, \mathcal{M}, c$) \land PALS($\psi_2, \mathcal{M}, c$) 
12: end if 
13: if $\phi$ is in form of $K_j \psi$ then 
14: \> return $\land_{c'}$PALS($\psi, \mathcal{M}, c'$); $\forall c' \in R_j(c)$ 
15: end if 
16: if $\phi$ is in form of $DA \psi$ then 
17: \> \> $R_A := \bigcap_{i \in A} R_i$ 
18: \> \> return $\land_{c'}$PALS($\psi, \mathcal{M}, c'$); $\forall c' \in R_A(c)$ 
19: end if 
20: if $\phi$ is in form of $[\psi_2][\psi_1]$ then 
21: \> if $\neg$PALS($\psi_2, \mathcal{M}, c$) then 
22: \>	ext{return true} 
23: else 
24: \>	ext{return PALS($\psi_1, \text{SFKE}($M, $\psi_2$), $c$)} 
25: end if 
26: end if 
27: return false 
28: end function

4 Linear Temporal Public Announcement Logic

In this section, an LTL extension of PAL is introduced to extract the knowledge of data stream inputs. It employs knowledge collected by classifiers in single-frame inputs and puts them in a hierarchical order.

Definition 4.1 (Language of LTPAL) We introduce an LTL extension of PAL by the following grammar in BNF:

$$
\phi ::= p \mid \neg \phi \mid (\phi \land \phi) \mid K_j \phi \mid DA \phi \mid [\phi] \phi
$$

$$
\Phi ::= \phi \mid (\neg \Phi) \mid (\Phi \land \Phi) \mid X \Phi \mid [\Phi U \Phi].
$$

Here, the temporal operators are $X \Phi$ (in the neXt data frame $\Phi$ must be true) and $[\Phi U \Phi]$ ($\Psi$ must remain true $\text{U}n\text{til} \Phi$ becomes true).

To define the Kripke semantic of this logic, assume that $M_0 = (\{c_00\}, R_00, \ldots, R_{0k}, V_0)$, $M_1 = (W_1, R_{10}, \ldots, R_{1k}, V_1), \ldots, M_{n-1} = (W_{n-1}, R_{(n-1)0}, \ldots, R_{(n-1)k}, V_{n-1})$, and $M_n = (\{c_{n0}\}, R_{n0}, \ldots, R_{nk}, V_n)$ are PAL models, where $W_i$’s are mutually disjoint sets, and $V_0(c_00) = V_n(c_{n0}) = \emptyset$. 
We build a new model $\mathcal{T}S = (S, R, s^0, s^{-1}, \to, L)$, known as a transition system, in which $S = \bigcup_{i=0}^{n} W_i$ is the set of states, $R = \{R_{ij} \mid 0 \leq i \leq n, 0 \leq j \leq k\}$, $s^0 = c_0$, and $s^{-1} = c_n$ are initial and final states, $\to = W_k \times W_{k+1}$, $0 \leq i < n$ are transition relations, $\equiv = \bigcup_{0 \leq k < n} \to^k$, $(c_{ki}, c_{(k+1)j}) \in \equiv^k$ is denoted by $\rightarrow_{ij}$, and the labeling function $L : S \to 2^{Prop}$, which are assigned propositional letters to states, is defined by $L(c_{ij}) = V_i(c_{ij})$, for each $c_{ij} \in W_i$. To extend the labeling function to all LTPAL formulas, we first fix the following notations.

An executive path $\pi_{TS}^{i\ldots j}$ in the model $\mathcal{T}S$ is a sequence of worlds $c_i c_{i+1} \ldots c_j$, where $c_i \rightarrow c_{i+1} \rightarrow \ldots \rightarrow c_j$ for $c_k \in W_k$, $i \leq k \leq j$. An executive path $\pi_{TS}^{i\ldots j}$ is called total and denoted by $\pi_{TS}$, if $c_i = s^0$ and $c_j = s^{-1}$. The set of all total executive paths will be denoted by $\Pi_{TS}$. Next, for $\phi \in$ PAL and $\Phi, \Phi_1, \Phi_2 \in$ LTPAL, we define:

- $\mathcal{T}S, \pi_{TS}^{i\ldots j} \models \phi$ if $M_i, c_i \models \phi$,
- $\mathcal{T}S, \pi_{TS}^{i\ldots j} \models X\phi$ if $\mathcal{T}S, \pi_{TS}^{i+1\ldots j} \models \phi$,
- $\mathcal{T}S, \pi_{TS}^{i\ldots j} \models \Phi_1 U \Phi_2$ if there exists $m, i \leq m \leq j$, $\mathcal{T}S, \pi_{TS}^{m\ldots j} \models \Phi_2$, and for all $k, i \leq k < m$, we have $\mathcal{T}S, \pi_{TS}^{i\ldots k} \models \Phi_1$.

In the first clause above, we consider the transition system TS over PAL formulas simply as a Kripke model. Moreover, Algorithm 5 is developed to create a transition system from PAL models. We remark that in Algorithm 5, to have the same beginning and end in calculating the paths, two extra models are added, in lines 3–12 and lines 23–29. Note that these two models do not affect the satisfaction of the input formula.

Temporal formulas can be investigated by the function TEMS developed in Algorithm 6. Here, the first and last Kripke models should be excluded. In other words, path $\pi_{TS}^{i\ldots j}$ in function $\text{TEMS}(\Phi, \mathcal{T}S, \pi_{TS}^{i\ldots j})$ should not start or end in the first and last Kripke models of the transition system. For example, for a transition system $\mathcal{T}S$, the investigation of formula $\Phi$ over a total path $\pi_{TS}^{0\ldots n}$ would be $\text{TEMS}(\Phi, \mathcal{T}S, \pi_{TS}^{1\ldots n-1})$. Let $n$ as the number of Kripke models in the transition system, $DeN_{\Phi}$ as the number of operators $\neg$, $\land$, and $X$ in the formula $\Phi$; and $DeU_{\Phi}$ be the number of operators $U$ in formula $\Phi$, the time complexity of Algorithm 6 would be $O(n^{DeU_{\Phi}} \times DeN_{\Phi})$.

Other temporal operators could be driven from the two operators of next ($X\Phi$), and until ($U\Phi$) in the following way:

- $F\Phi \equiv (\top \land \Phi)$,
- $(\Phi \land \Psi) \equiv \neg(\neg\Phi \land \neg\Psi)$,
- $(\Phi \land \Psi) \equiv (\neg\Phi \lor \neg\Psi)$,
- $G\Phi \equiv (\top \land \Phi)$.

The intended meaning of future ($F\Phi$) is “eventually $\Phi$ becomes true”, global ($G\Phi$) is “$\Phi$ must remain true forever”, release ($\Phi \land \Psi$) is “$\Psi$ remains true until and including when $\Phi$ becomes true, if $\Phi$ never becomes true, $\Psi$ always remains true”, and weak until ($\Phi \land \Psi$) is “$\Phi$ has to remain true at least until $\Psi$; if $\Phi$ never holds, $\Phi$ must always remain true” [13, 15].

**Theorem 4.2** Let $\mathcal{T}S = (S, R_1, \ldots, R_n, s^0, s^{-1}, \to, L)$ be a transition system and $\Phi$ an LTPAL formula. Then for $\mathcal{T}S, \pi_{TS}^{i\ldots j} \in \Pi$, we have $\mathcal{T}S, \pi_{TS}^{i\ldots j} \models \Phi$ iff $\text{TEMS}(\Phi, \mathcal{T}S, \pi_{TS}^{i\ldots j})$. 

 Springer
Algorithm 5 The Time-Series Transition System (TSTS) function will create the transition system by the given input information

Let $C$ as the set of classifiers, $X$ the time-series input data of size $k$ ($X = [x_1, \ldots, x_k]$), and $\rho$ the predefined property set. We also use the MASKS function developed in [5].

1: function TSTS($C$, $X$, $\rho$)
2: $k := \text{size}(X)$
3: $S := \{c_0\}$
4: $W := \{c_0\}$
5: $R_1, \ldots, R_n := \{(c_0, c_0)\}$
6: $s_0 := c_0$
7: $s^{-1} := c_0$
8: $\rightarrow := \emptyset$
9: $V(c_0) := \emptyset$
10: $L(c_0) := \emptyset$
11: $M := (W, R_1, \ldots, R_n, V)$
12: $T_S := (S, R_1, \ldots, R_n, s^0, s^{-1}, \rightarrow, L)$
13: for all $x_0$ in $X$ do
14: \hspace{1cm} $\_ := \text{MASKS}(C, x_0, \rho)$
15: \hspace{1cm} $\Delta := \text{Kripke model } M' = (W', R'_1, \ldots, R'_n, V')$
16: \hspace{1cm} $S := S \cup W'$
17: \hspace{1cm} $\forall_i R_i := R_i \cup R'_i$
18: \hspace{1cm} for all $c \in W$ and $c' \in W'$ do
19: \hspace{2cm} $\rightarrow := \rightarrow \cup \{(x, c')\}$
20: \hspace{1cm} end for
21: \hspace{1cm} $M := M'$
22: end for
23: $W := \{c_{k+1}\}$
24: $S := S \cup W$
25: $\forall_i R_i := R_i \cup \{(c_{k+1}, c_{k+1})\}$
26: $s^{-1} := c_{k+1}$
27: for all $c \in W$ and $c' \in W'$ do
28: \hspace{1cm} $\rightarrow := \rightarrow \cup \{(x, c')\}$
29: end for
30: return $T_S$
31: end function

Proof We only prove the if direction; the other direction is proved similarly. The proof is by induction on the complexity of $\Phi$.

For any path formula in which the path length is lower than zero, lines 3 and 4 would be executed, and false will be returned. For any PAL formula, lines 6–8 would be executed. Similar to the semantics of LTPAL, we will check whether $\Phi$ is in $PALS(\Phi, M, c)$ or not. Here, $M$ is the $i$-th Kripke model of the transition system $T_S$.

Suppose that $T_S, \pi_{TS}^i \models \Phi$, and $\Phi \in$LTPAL is not a PAL formula. Thus, $\Phi$ would be in the form of $\neg\Phi_1, \Phi_1 \wedge \Phi_2, X\Phi_1$, or $\Phi_1 U \Phi_2$.

- If $\Phi \equiv \neg\Phi_1$, then the if condition at line 11 would be satisfied, and line 12 would be executed. In this line, we return the negation of TEMS($\Phi_1, T_S, \pi_{TS}^i$). Therefore, by induction, TEMS($\Phi_1, T_S, \pi_{TS}^i$) if $T_S, \pi_{TS}^i \models \Phi_1$. Consequently, TEMS($\Phi, T_S, \pi_{TS}^i$) if $T_S, \pi_{TS}^i \models \Phi$. 

 Springer
Algorithm 6 The TEMporal Satisfaction function (TEMS) will investigate the satisfaction of LTPAL formulas

Let $\Phi$ as the LTL formula, $\mathcal{T}S$ the transition system, and $c$ a world

1: function TEMS($\Phi$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$)
2: 
3: if $i > j$ then
4: return false
5: end if
6: if $\Phi$ is a PAL formula then
7: $\mathcal{M} :=$ the $i$-th Kripke model of $\mathcal{T}S$
8: return PALS($\Phi$, $\mathcal{M}$, $c_i$)
9: end if
10: if $\Phi$ is in form of $\neg \Psi$ then
11: return $\neg$ TEMS($\Psi$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$)
12: end if
13: if $\Phi$ is in form of $\Psi_1 \land \Psi_2$ then
14: return TEMS($\Psi_1$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$) $\land$ TEMS($\Psi_2$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$)
15: end if
16: if $\Phi$ is in form of $X \Psi$ then
17: return TEMS($\Psi$, $\mathcal{T}S$, $\pi_{TS}^{i+1,j}$)
18: end if
19: if $\Phi$ is in form of $\Psi_1 U \Psi_2$ then
20: if TEMS($\Psi_2$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$) then
21: return true
22: end if
23: while TEMS($\Psi_1$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$) do
24: $i := i + 1$
25: if TEMS($\Psi_2$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$) then
26: return true
27: end if
28: end while
29: end if
30: return false
31: end function

- If $\Phi \equiv \Phi_1 \land \Phi_2$, then the if condition at line 14 would be satisfied, and line 15 would be executed. At this line, we return the conjunction of TEMS($\Phi_1$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$) and TEMS($\Phi_2$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$). Therefore, by induction, TEMS($\Phi_1$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$) if $\mathcal{T}S$, $\pi_{TS}^{i,j} \models \Phi_1$ and TEMS($\Phi_2$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$) if $\mathcal{T}S$, $\pi_{TS}^{i,j} \models \Phi_2$. Consequently, TEMS($\Phi$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$) if $\mathcal{T}S$, $\pi_{TS}^{i,j} \models \Phi$.
- If $\Phi \equiv X \Phi_1$, then the if condition at line 17 would be satisfied, and line 18 would be executed. In this line, TEMS($\Phi_1$, $\mathcal{T}S$, $\pi_{TS}^{i+1,j}$) would be calculated; true would be returned if and only if for the next state of the path $\pi_{TS}^{i,j}$, which is $\pi_{TS}^{i+1,j}$, $\Phi_1$ is true. By induction, we have TEMS($\Phi_1$, $\mathcal{T}S$, $\pi_{TS}^{i+1,j}$) if $\mathcal{T}S$, $\pi_{TS}^{i+1,j} \models \Phi_1$. By definition, we have $\mathcal{T}S$, $\pi_{TS}^{i+1,j} \models \Phi$ if $\mathcal{T}S$, $\pi_{TS}^{i,j} \models X \Phi$. Consequently, TEMS($\Phi$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$) if $\mathcal{T}S$, $\pi_{TS}^{i,j} \models \Phi$.
- If $\Phi \equiv \Phi_1 U \Phi_2$, then the if condition at line 20 would be satisfied, and lines 21–29 would be executed. In line 21, first, we check if TEMS($\Phi_2$, $\mathcal{T}S$, $\pi_{TS}^{i,j}$)
is true. By definition, we have: \( T S, \pi_{T S}^{i..j} \models \Phi_1 U \Phi_2 \), if there exists \( m, i \leq m \leq j \), \( T S, \pi_{T S}^{m..j} \models \Phi_2 \), and for all \( k, i \leq k < m \), we have \( T S, \pi_{T S}^{i..k} \models \Phi_1 \). Hence, if \( m = i \), TEMS(\( \Phi_1, T S, \pi_{T S}^{i..j} \)) would be true. Otherwise, by definition, TEMS(\( \Phi_1, T S, \pi_{T S}^{i..j} \)) must be true over the path while satisfying TEMS(\( \Phi_2, T S, \pi_{T S}^{i..j} \)). If TEMS(\( \Phi_2, T S, \pi_{T S}^{i..j} \)) is never satisfied while TEMS(\( \Phi_1, T S, \pi_{T S}^{i..j} \)) is true, false should be returned (line 31). By induction, we have TEMS(\( \Phi_1, T S, \pi_{T S}^{i..j} \)) if \( T S, \pi_{T S}^{i..k} \models \Phi_1 \) and TEMS(\( \Phi_2, T S, \pi_{T S}^{i..j} \)) if \( T S, \pi_{T S}^{i..k} \models \Phi_2 \). Consequently, TEMS(\( \Phi_1, T S, \pi_{T S}^{i..j} \)) if \( T S, \pi_{T S}^{i..j} \models \Phi \). \( \square \)

We conclude the paper by giving a formal definition of verification, possibility, and missing information in a transition system. Thus, the following notations are introduced first.

**Notation 4.3** Let \( \psi, \phi \in \text{PAL}, \Phi, \Phi_1, \Phi_2 \in \text{LTPAL}, \Phi_1', \Phi_2' \in \text{LTPAL/PAL}, O_1 = \{\land, \lor\}, O_2 = \{U, R\}, O = O_1 \cup O_2, f(\land) = \lor, f(\lor) = \land, f(U) = R, f(R) = U \). For \( \Omega \in \{DA, \neg DA, K_i, \neg K_i, [\psi]DA, [\psi]\neg DA, [\psi]K_i, [\psi]\neg K_i\} \) we define \( \Omega \Phi \) as follows:

\[
\Omega \Phi = \begin{cases} 
\Omega \Phi_1 & \text{iff } \Phi = \Phi_1, \\
X \Omega \Phi_1 & \text{iff } \Phi = X \Phi_1, \\
X \neg \Phi_1 & \text{iff } \Phi = \neg X \Phi_1, \\
\Omega \Phi_1 O \Omega \Phi_2' & \text{iff } \Phi = \Phi_1 O \Phi_2', \\
\Omega (\neg \Phi_1 f (O_1) \neg \Phi_2') & \text{iff } \Phi = \neg (\Phi_1' O_1 \Phi_2'), \\
\Omega (\neg \Phi_1 f (O_2) \neg \Phi_2) & \text{iff } \Phi = \neg (\Phi_1 O_2 \Phi_2). 
\end{cases}
\]

(1)

Now, for any transition system \( T S \) and \( \Phi \in \text{LTPAL} \), we define verification of \( \Phi \) in \( T S \) formula as follows:

1. \( \Phi \) is verified in \( T S \) for the group \( A \) of classifiers exactly when we have \( T S, \pi_{T S} \models D_A \Phi \).
2. \( \Phi \) is possible in \( T S \) for the group \( A \) of classifiers exactly when we have \( T S, \pi_{T S} \models \neg D_A \neg \Phi \).
3. A PAL formula \( \psi \) is called verified-missing information in \( T S \) for a formula \( \Phi \) in group \( A \) of classifiers exactly when we have \( T S, \pi_{T S} \not\models D_A \Phi \) and \( T S, \pi_{T S} \models [\psi](D_A \Phi) \).
4. A PAL formula \( \psi \) is called possible-missing information in \( T S \) for a formula \( \Phi \) in group \( A \) of classifiers exactly when we have \( T S, \pi_{T S} \not\models \neg D_A \neg \Phi \) and \( T S, \pi_{T S} \models [\psi](\neg D_A \neg \Phi) \).

**Example** Assume a one-bed medical operating room, which is monitored by a camera. An operation is carried out on three animals, which may be dogs, wolves, or foxes, denoted by \( d, w, \) and \( f \). During these operations, someone revealed that the sterilization system did not work; we know that dogs have the flu. It could spread from dogs to wolves and from wolves to foxes if they go to the operating room after each other. No one knows the order of the operation. Thus, they fed three respective camera images.
Fig. 1 $R_{i,1}$ and $R_{i,2}$ are green and blue lines, respectively

$X = \{x_1, x_2, x_3\}$ of operations into two classifiers $A = \{1, 2\}$ to find out whether the fox contracted the flu or not. The first classifier doubts fox and dog for the first image, wolf and dog for the second, and fox and wolf for the third. The second classifier is uncertain about the wolf and dog for the first image, about the fox and wolf for the second, and about the wolf and dog for the third. We will apply our model to this scenario. Let $(x_i, d)$, $(x_i, w)$, and $(x_i, f)$ as atomic formulas ($d_i$, $w_i$ and $f_i$ in brief), demonstrating the existences of dog, wolf, and fox for the $i$-th image, respectively. By Algorithm 2, Kripke models $\mathcal{M}_i = (W_i, R_{i,1}, R_{i,2}, V_i)$, for $i \in \{1, 2, 3\}$, are shown in Fig. 1, where $V_i(c) = \{c\}$, for $c \in \{d_i, w_i, f_i\}$ and $V_i(\bar{c}_i) = \{d_i, w_i, f_i\}$.

The resulting transition system $\mathcal{T}S = (S, R, s^0, s^{-1}, \rightarrow, L)$ of $\mathcal{M}_i$’s, by Algorithm 5, is depicted in Fig. 2, where $L = V_1 \cup V_2 \cup V_3$ and $L(s^0) = L(s^{-1}) = \emptyset$. 

$\mathcal{M}_1$ $\mathcal{M}_2$ $\mathcal{M}_3$
Next, we would like to check whether any fox contracted the flu or not, which can be expressed in LTPAL by $\Phi = d_1 \land X w_2 \land X X f_3$. By Algorithm 6, $TS, \pi_{TS} \not\models D_A \Phi$, which means that the system does not ensure that a fox contracts the flu (it is not verified). Also, $TS, \pi_{TS} \not\models \neg D_A \neg \Phi$ means that a fox did not contract the flu (it is impossible).

Next, we check the impact of the collaboration of agents and announcements. For the former, assume that, in the beginning, wolves also had the flu which can be expressed by $\Phi' = F(w_1 \land X f_2) \lor F(w_2 \land X f_3)$. Then, $TS, \pi_{TS} \not\models \neg D_A \neg \Phi'$, which means that from group A’s perspective, contracting the flu is impossible for a fox. Nevertheless, no single agent knows that the flu is impossible for a fox. Since $TS, \pi_{TS} \models \neg D_A \neg \Phi'$, for $A_i = \{i\}$. For the latter, assume that someone announced the second operation was done on a wolf. In this case, we have $TS, \pi_{TS} \not\models [w_2] \neg D_{A_2} \neg \Phi'$. It means that the second agent could ensure that the flu is impossible for a fox when $w_2$ is announced.

5 Conclusions and Future Work

In this study, an approach was designed to formalize acquired knowledge by any classification. Next, by introducing LTPAL, an extension of PAL, the flow of knowledge in data streams was modeled. This model presents acquired knowledge in a transition system structure. It can examine the satisfaction of properties (LTPAL formulas) to solve classifiers’ verification problems. Consequently, we investigated the reliability of the answers. As an application, we explained a scenario in which “natural language translations to LTPAL” should be verified. Moreover, this model can answer the question: “which missing knowledge could lead the system to the correct answer?”. Thus, the developed model was suitable for interpreting and verifying multi-classifiers that classify the data streams.

We will extend the model to consider the probabilities. Then, we can find the most probable scenario for each data stream to assist classifiers. Besides, we will define notations to model multi-object classifiers in a unified way. For example, in an input image, multiple objects could be distinguished by their location in the picture.

Author Contributions All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no financial or proprietary interests in any material discussed in this article.

Funding The authors have no relevant financial or non-financial interests to disclose.

Declarations

Conflict of Interest The authors have no competing interests to declare that are relevant to the content of this article.

Appendix A: The Developed Tool

This section describes the developed tool, which is done in the Python programming language. This tool aims to create an LTPAL model for data stream inputs. To do
this, first, we collect all knowledge gained from classifiers for specific input. This knowledge is “predicted output classes” collected by group $A$ of classifiers. As mentioned, the knowledge is provided from a data stream so that classifiers would provide output classes for each data frame of the stream. Here, a Kripke model would be developed based on the provided knowledge for output classes of each data frame. Consequently, possible worlds of the Kripke model would be calculated. Thus far, for each data frame of the data stream, a Kripke model was provided. Using these Kripke models and adding them in sequential order, we could create a transition system as we described in Sect. 4. Now, LTPAL formulas could be investigated over paths of the transition system. Due to the theory provided in Sect. 4, the condition of an LTPAL formula $\Phi$ (verified, possible, verified-missing, or possible-missing) over path $\pi_{T\Sigma}$ of the transition system is investigated.

A.1 The Atomic Formula

As it is assumed, the classifiers have a specified output domain. It means that for a group of classifiers $A$, the aggregated knowledge would be considered as the group’s output domain $C$. The output domain defines all possible outcomes of classifiers for every input. Here, a pair $(x, c)$ for each class $c \in C$ from the output domain of a specific input $x$ will be stored as the atomic formula. We created a Python class, namely “AtomicFormula” with the property “inputData”, which is referred to the specific data frame, and the property “name”, which is referred to the name of the output class. We created an instance for each pair of input $x$ and output class $c$.

A.2 The Kripke Model

In this section, we will describe the Kripke model’s development in the tool. It should be considered that a PAL Kripke model is a tuple $M = (W, R_1, \ldots, R_n, V)$. Therefore, we developed a python class called “KripkeModel”, including $W, R_i; 1 \leq i \leq n$, and $V$ as its properties. To create a proper Python class instance, the following steps should be taken. First, all output classes are collected for a specific input data frame. Second, the intersection of the classes provided by classifiers as the possible output classes is calculated (see Algorithm 2). The cardinality power set of such output classes is the number of possible worlds in this step. Here, the initial Kripke model is created as follows:

- Let the cardinality power set of output classes be $n$,
- $W = \{c_1, \ldots, c_n, \bar{c}\}$,
- $R = \{(i, j) \mid i, j \in \{c_1, \ldots, c_n, \bar{c}\}\}$,
- $V(c_i) = \{(x, c_i)\}, V(\bar{c}) = \{(x, c_i) \mid i \in \{1, \ldots, n\}\}$, $(x, c_i)$ is from the “Atomic-Formula” class.

After creating the Kripke model $M$, the PAL formula $\varphi$ could be investigated for each world. Herein, for all provided PAL formulas, and all worlds $c \in W$, if $M, c \not\models \varphi$, the world is considered impossible and removed from the Kripke model $M$. In this
process, we also remove all relations with one end in $c$ and all evaluation functions $V(c)$. Therefore, the created Kripke model is reduced to $M^\varphi$.

A.3 The Transition System

In this step, a list of Kripke models was provided, each related to a data frame. Therefore, to create the transitions system, first, we created a Python class, named “TransitionSystem”. It contains properties $S$, $R$, $s^0$, $s^{-1}$, $\rightarrow$, and $L$, for a transition system $TS$. This Python class contains a Python method “add_kripke” that gets a “KripkeModel” object as input and appends it to the end of the “TransitionSystem”. By calling this Python method for all Kripke models, the transition system is created in sequential order. The next step is to investigate the LTPAL formula $\varphi$ in this transition system.

Appendix B: Extended Data: How to Use the LTPAL Tool

We developed the LTPAL tool to verify the properties of multiple classifiers. Here, properties appear in the form of LTPAL formulas, which should be investigated for single-framed data (i.e., images) or data streams (i.e., videos).

B.1 The Input

As stated, the tool could apply to an ensemble of classifiers. Therefore, one should collect suitable classifiers. Then, each input (and its neighborhoods) should be fed into the classifiers. The output is a list of detected objects for each data frame, so there is a list of lists of objects for each data stream. The input should consist of:

- LTPAL formulas (properties),
- PAL formulas (arbitrary constraints),
- classifiers’ output class domain (C),
- number of frames (1 for single data frame),
- classifiers’ ids,
- for each classifier, predictions for each frame should be written.

We provide Sample inputs on the tool’s webpage (“classifiersPredictions.json” and “subsetDict.json” files) for more information.

B.2 The Output

The expected output is a log file, including:

- Kripke model of each data frame,
- evaluation of each property over path $\pi_{TS}$,
- defining whether each formula is a verified answer or a possible answer,
- the transition system.
Sample output file is provided on the tool’s webpage (“output_log_file.log” and “result.json” files).

B.3 The Execution

After the configuration of the input files, “MASKS.py” is executed using the Python3.

The file “classifiersPredictions.json” contains all LTPAL and PAL formulas, output classes, number of frames, classifiers’ ids, and each frame’s extracted information for each classifier. Sample “classifiersPredictions.json”:

```json
{
  "formulas_LTPAL": ["X_iU_i(cat, chair)", "X_i&(K_i
~&(cat, dog), cat)", "X_ichair", "X_idog"],
  "formulas_PAL": ["&(K_i~&(cat, dog), cat)"],
  "allClasses": ["cat", "dog", "chair"],
  "number_of_frames": 2,
  "classifiers_ids": ["1", "2"],
  "1": [
    [{"name": "cat"}, {"name": "dog"}, {"name": "chair"}],
    [{"name": "cat"}, {"name": "dog"}]
  ],
  "2": [
    [{"name": "cat"}, {"name": "chair"}],
    [{"name": "cat"}]
  ]
}
```

The file “result.json” is created after the execution, which contains the created transition system. Sample “result.json”:

```json
{
  "S": [[[0, 0]], [[1, 2], [1, 3]], [[2, 1]], [[3, -1]]],
  "R": [[[0, 0], [0, 0]], [[[1, 2], [1, 2]], [[1, 2], [1, 3]], [[1, 3], [1, 2]], [[1, 3], [1, 3]], [[2, 1], [2, 1]], [[3, -1], [3, -1]]],
  "S_0": 0,
  "S_1": -1,
  "Arrow": [[[0, 0], [1, 2]], [[0, 0], [1, 3]], [[1, 2], [2, 1]], [[1, 3], [2, 1]], [[2, 1], [3, -1]]],
  "L": {
    "0_0": [], "1_2": [{"name": "cat"}], "1_3": [{"name": "chair"}], {"name": "cat"}], "2_1": [{"name": "cat"}], "3_-1": []
  }
}
```
The execution would also provide a file. This file contains information extracted from the inputs. Each frame provides each classifier’s outputs, the common information, and the Kripke model for that frame. Then, all execution paths are provided. Next, for each input formula, the tool provides the state of the formula (i.e., verified/possible). Consequently, the transition system is provided. Sample “output_log_file.log”:

```plaintext
MASKS started
---------------------------------->
___Creating Kripke Model for frame: 0
________
_______classifiers_prediction______________
[[{'name': 'cat'}, {'name': 'dog'}, {'name': 'chair'}], [{'name': 'cat'}, {'name': 'chair'}]]
________ArrayOfOutputClasses______________
[[cat), (dog), (chair)], [(cat), (chair)]
______IntersectedArrayOfOutputClasses_______
[(cat), (chair)], [(cat), (chair)]
The input formula: &K_i ~(cat, dog), cat
removed list of worlds: 1 with names ['chair'] in the kripke Model of frame: 0
_______kripke_______________ for frame
number: 0

\[
W = [2, 3], \\
R = [(2, 2), (2, 3), (3, 2), (3, 3)], \\
V = \{2: \{"name": "cat"\}, \\
3: \{"name": "chair"\}, \{"name": "cat"\}\}
\]

___Creating Kripke Model for frame: 1
________
_______classifiers_prediction______________
[[{'name': 'cat'}, {'name': 'dog'}], [{'name': 'cat'}]]
________ArrayOfOutputClasses______________
[(cat), (dog)], [(cat)]
______IntersectedArrayOfOutputClasses_______
[(cat)], [(cat)]
no world removed by PAL formula for frame
number: 1
_______kripke_______________ for frame
number: 1

\[
W = [1], \\
R = [(1, 1)], \\
V = \{1: \{"name": "cat"\}\}
\]
```
The tool is currently under development, which can be found at https://github.com/iuwa/LTPAL.

References

1. Anne Hendricks, L., Wang, O., Shechtman, E., Sivic, J., Darrell, T., Russell, B.: Localizing moments in video with natural language. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5803–5812 (2017)
2. Bauer, L., Wang, Y., Bansal, M.: Commonsense for generative multi-hop question answering tasks. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 4220–4230. Association for Computational Linguistics, Brussels, Belgium (2018). https://doi.org/10.18653/v1/D18-1454 . https://www.aclweb.org/anthology/D18-1454
3. Bjesse, P.: What is formal verification? ACM SIGDA Newsletter. 35(24), 1 (2005)
4. de Leng, D., Heintz, F.: Approximate stream reasoning with metric temporal logic under uncertainty. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 2760–2767 (2019)
5. Dehkordi, A.H., Alizadeh, M., Movaghar, A.: Meet MASKS: a novel Multi-Classifier’s verification approach (2020). https://doi.org/10.48550/ARXIV.2007.10090. https://arxiv.org/abs/2007.10090
6. Fallah, H., Didehvar, F., Rahmati, F.: Approximation algorithms for the load-balanced capacitated vehicle routing problem. Bull. Iran. Math. Soc. 47(4), 1261–1288 (2021)
7. Fong, B., Speranzon, A., Spivak, D.I.: Temporal landscapes: A graphical temporal logic for reasoning (2019). arXiv preprint arXiv:1904.01081
8. Gao, J., Sun, C., Yang, Z., Nevatia, R.: Tall: Temporal activity localization via language query. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5267–5275 (2017)
9. Gerth, R., Peled, D., Vardi, M.Y., Wolper, P.: Simple on-the-fly automatic verification of linear temporal logic. In: International Conference on Protocol Specification, Testing and Verification, pp. 3–18 (1995). Springer
10. Goldman, R.P., Charniak, E.: Probabilistic text understanding. Stat. Comput. 2(2), 105–114 (1992)
11. Golmohamadian, M., Zahedi, M.M.: The language of epidemic. Bull. Iran. Math. Soc. 48(5), 2105–2123 (2022)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
13. Hossain, A., Laroussinie, F.: From quantified ctl to qbf (2019). arXiv preprint arXiv:1906.10005
14. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) Computer Aided Verification, pp. 3–29. Springer, Cham (2017)
15. Kwiatkowska, M., Lomuscio, A., Qu, H.: Parallel model checking for temporal epistemic logic. In: Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence, pp. 543–548 (2010)
16. Ma, K., Francis, J., Lu, Q., Nyberg, E., Oltramari, A.: Towards generalizable neuro-symbolic systems for commonsense question answering (2019). arXiv preprint arXiv:1910.14087
17. Nilsson, N.J.: The Quest for Artificial Intelligence, 1st edn. Cambridge University Press, Cambridge (2009)
18. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
19. Russell, S., Norvig, P.: Artificial intelligence: a modern approach (2002)
20. Sap, M., Le Bras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, H., Roof, B., Smith, N.A., Choi, Y.: Atomic: An atlas of machine commonsense for if-then reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 3027–3035 (2019)
21. Speer, R., Lowry-Duda, J.: Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge (2017). arXiv preprint arXiv:1704.03560
22. Winston, PH.: Artificial Intelligence, vol. 34, 3rd edn., pp. 167–339. Addison-Wesley, Reading (1992)
23. Yang, P., Zhang, Z., Zhou, B.B., Zomaya, A.Y.: Sample subset optimization for classifying imbalanced biological data. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 333–344. Springer, Berlin, Heidelberg (2011)
24. Yi, K., Gan, C., Li, Y., Kohli, P., Wu, J., Torralba, A., Tenenbaum, J.B.: Clever: Collision events for video representation and reasoning (2019). arXiv preprint arXiv:1910.01442

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.