Final Report for Program Synthesis (CS 6172)

Inferring Temporal Logic Specifications for Robot-Assisted Feeding in Social Dining Settings

Janko Ondras (jo339)
December 6, 2021

***Abstract*— In this project, I infer temporal logic specifications of human eating behaviors during social dining and I specifically focus on prediction of bite transfer timing. First, I present a novel dataset containing audio-visual recordings of people eating together. Using high-level features from these observations I then infer Signal Temporal Logic (STL) specifications of appropriate bite transfer timings. Such formal rules are interpretable and can be used to drive behavior of an assistive robot that feeds a person with disabilities.**

I. INTRODUCTION

In 2010, more than 975 million people worldwide had a disability [1]. Almost 20% of them needed assistance with activities of daily living or instrumental activities of daily living. One of the most important such activities is feeding which is time-consuming for the caregiver and challenging for the care recipient (patient) to accept socially [2]. Even though there exist several automated robotic feeding systems, they require the user to trigger the bite transfer which puts burden on the user and is challenging in social settings. Moreover, in a social dining scenario the timing of a bite transfer is likely to depend on the group dynamics and multimodal social cues.

To address this problem, I collected an audio-visual dataset capturing human eating behaviors in groups and using these observations, I automatically inferred temporal logic specifications of suitable bite transfer timings. There are several advantages of the formal methods approach: 1) it learns rules that offer interpretable explanations for appropriate bite transfer timings as compared to black-box models; 2) it does not require lots of training data as compared to neural network-based models for time series data, such as [3]; and 3) comparing to model-based machine learning methods applied to time series data, such as [4], it does not necessarily require a model of the analyzed system which is often unknown. Signal Temporal Logic (STL) is an especially suitable formalism for this problem as it allows reasoning about properties of dense-time real-valued signals which fits well with the high-level features that can be extracted from the observed audio-visual data. As a last step, the learned rules can be used to drive a behavior of an assistive feeding robot.

The main contributions of this work can be summarized as follows.

- Novel audio-visual dataset capturing triadic human interactions in a social dining setting.
- Application of STL specification inference to model human/multi-human interactions in a social dining scenario.
- Extensive experiments comparing misclassification rates and training times across various hyperparameters and settings.
- Finding that the target participant's head rotation features are much more informative for the prediction of bite transfer timings for this participant than these kind of features from the other two participants.

The rest of this report is organized as follows. Section II reviews the related work in the field, Section III defines the STL formalism, and Section IV details the employed *LoTuS* system [5] for STL specification learning. The novel social dining dataset is presented in Section V and the methodology and performed experiments are described in Section VI. The obtained results are presented and discussed in Section VII. Section VIII summarizes this work and Section IX suggests directions for further research.

II. RELATED WORK

The field of STL specification/formulae learning/mining is very rich and can be reviewed in terms of the following characteristics. For a more general taxonomy of specification learning algorithms, I refer the reader to a recent survey on learning-based formal methods [6].

1) Types of system traces: Each system trace (a set of real-valued signals observed over a certain time interval) is assigned either a positive label (the corresponding trace satisfies the desired property) or a negative label (the corresponding trace violates the desired property). If both positive and negative examples of system traces are available, the STL formula learning problem can be cast as a supervised learning problem. The goal is to construct a discriminating STL formula that acts as a binary classifier and can correctly assign a trace to one of the two classes such that the misclassification rate (MCR) is minimized. Many works [5]–[15] addressed this problem using various approaches. For instance, the *TempLogIn* algorithm proposed by Kong et al. [7] considered a special fragment of STL, organized the formulae in a directed acyclic graph, and performed a breadth-first search. Bartocci et al. [8] and Bufo et al. [9]

learned a generative model for each class and then searched for a formula that best discriminates between the outputs of the two models. Other works [5], [12], [14], [16], [17] proposed decision tree-based methods where the test associated with a tree node corresponds to a simple formula tuned from a set of primitive formulae.

However, sometimes only the positive examples of system behavior are available which presents a challenge not to infer overly general formulae. Only few works addressed this problem using STL [15], [18]–[21] or using other kinds of temporal logic [22]–[25]. For example, the authors of the *TeLex* system [18], [19] defined a new tightness metric to find STL formula parameters preventing over-generalization while satisfying all the example traces.

2) Template-based vs. template-free: Many works [18]–[20], [26]–[31] assumed that the structure of the STL formula is provided as a parametric STL (PSTL) formula template. For example, a domain expert provides a formula template such as *"Within the next τ time-steps the car velocity must be less than v_m."* The specification learning problem then becomes an optimization problem where the goal is to find optimal parameters (τ and v_m in the above example) of the PSTL formula subject to an objective function (e.g., in terms of the formula robustness degree [26]). This approach however relies on domain expert knowledge and it is thus not well suited for specifying behaviors of unexplored systems.

Other works [5], [7]–[10], [12], [13], [16], [17], [32] infer both the formula structure and the parameters. This is usually addressed in two steps: 1) learning the formula structure, and 2) optimizing the formula parameters. For example, the structure can be learned by exploring the directed acyclic graph [13], by constructing a decision tree [5], [12], or by using genetic algorithms [8]–[10]. Recently, Mohammadinejad et al. [11] took inspiration from syntax-guided synthesis (learning expressions from grammars) and proposed another technique based on systematic enumeration of STL formulae with heuristic pruning of equivalent formulae.

3) Offline, online, and active learning: In the offline learning scenario, all the training data is provided upfront and the final STL formula is constructed. In contrast, the online learning setting allows the STL formula to be refined as new training examples arrive over time which might be beneficial for model deployment. Most previous works considered only the offline learning problem and only several works [12], [33] presented both offline and online algorithms. There are also active learning methods [26]–[29], [34] that can query the system for new signals to explore the boundaries of legal behavior.

This work falls into the category of supervised binary classification since I obtained both positive and negative examples corresponding to appropriate and inappropriate bite transfer timings respectively. I considered a template-free approach that does not require strong presumptions about formulae structure because I dealt with a new kind of problem without any domain expert knowledge. For now, I addressed only the offline learning case. For these reasons, I decided to follow the decision tree-based approach of Bombara and Belta [5] and used their *LoTuS* system detailed in Section IV.

III. SIGNAL TEMPORAL LOGIC (STL)

Signal Temporal Logic (STL) [35] is a formalism to express properties of real-valued time-series data. It has been applied in various areas such as biology [36], robotics [37], analog circuits [38], and automotive systems [39]–[41].

The STL **syntax** is recursively defined as follows.

$$\phi \coloneqq \text{true} \mid f_i(t) \sim \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 U_{[a,b)} \phi_2$$

where ϕ is an STL formula, $f_i(t) \sim \mu$ is a predicate with

- an i-th signal feature $f_i(t) \in \mathbb{R}$ at time t where $i \in \{1, \ldots, n\}$ and n is the number of signal features,
- an order relation $\sim \in \{\leq, >\}$, and
- a real number $\mu \in \mathbb{R}$,

$a, b \in \mathbb{R}^+$ are timing bounds, and U is the bounded temporal operator *until*.

The STL **semantics** is defined over signals $f(t) \in \mathbb{R}^n$ at time t as follows.

$$\begin{aligned}
f(t) &\models \text{true} \Leftrightarrow \text{true} \\
f(t) &\models f_i(t) \sim \mu \Leftrightarrow f_i(t) \sim \mu \\
f(t) &\models \neg\phi \Leftrightarrow \neg(f(t) \models \phi) \\
f(t) &\models \phi_1 \wedge \phi_2 \Leftrightarrow (f(t) \models \phi_1) \wedge (f(t) \models \phi_2) \\
f(t) &\models \phi_1 U_{[a,b)} \phi_2 \Leftrightarrow \exists t_u \in [t+a, t+b).\ (f(t_u) \models \phi_2) \\
&\qquad \wedge (\forall t' \in [t, t_u).\ f(t') \models \phi_1)
\end{aligned}$$

An STL formula ϕ is satisfied by the signal f if and only if $f(0) \models \phi$. Two additional temporal operators *eventually* and *always* are often used and are defined, respectively, as $\diamond_{[a,b)}\phi \equiv \text{true}\ U_{[a,b)}\ \phi$ and $\square_{[a,b)}\phi \equiv \neg \diamond_{[a,b)} \neg\phi$.

Parametric Signal Temporal Logic (PSTL) [20] extends STL with *time* and *space* parameters that replace the time constants a, b and the constant μ, respectively. Every parameter assignment (also called valuation) of a PSTL formula produces one STL formula.

IV. STL LEARNING SYSTEM LOTUS

The *LoTuS* system by Bombara and Belta [5] is a decision tree-based framework implemented in Matlab for learning STL specifications of system behavior. Given a set of labeled system traces (including both positive and negative examples), the system infers an STL formula such that the misclassification rate (MCR) is minimized.

Since the optimal decision tree learning is NP-complete [42], a common approach is to make a greedy, locally best, choice at each node. The *LoTuS* system builds a binary decision tree such that each non-leaf node splits the traces into two children nodes according to a simple STL formula associated with that node. Each such an STL formula is tuned from a finite set of PSTL formulae called primitives. Two types of primitives are defined as follows.

- First-level primitives

$$P_1 = \{ \diamond_{[\tau_1,\tau_2]} (f_i(t) \sim \mu) \text{ or } \square_{[\tau_1,\tau_2]}(f_i(t) \sim \mu) \mid i \in \{1,\ldots,n\}, \sim \in \{\leq, >\}\}$$

  where n is the number of signal features and $\mu \in \mathbb{R}, \tau_1, \tau_2 \in \mathbb{R}_{\geq 0}$. $\tau_1 < \tau_2$ are parameters of the PSTL formulae in P_1.
- Second-level primitives

$$P_2 = \{\square_{[\tau_1,\tau_2]} \diamond_{[0,\tau_3]} (f_i(t) \sim \mu) \text{ or } \diamond_{[\tau_1,\tau_2]} \square_{[0,\tau_3]}(f_i(t) \sim \mu) \mid i \in \{1,\ldots,n\}, \sim \in \{\leq, >\}\}$$

  where n is the number of signal features and $\mu \in \mathbb{R}, \tau_1, \tau_2, \tau_3 \in \mathbb{R}_{\geq 0}$. $\tau_1 < \tau_2$ are parameters of the PSTL formulae in P_2.

The parameters in these primitive PSTL formulae need to be bounded and can be inferred from the input data. The best primitive for a node is chosen based on its impurity measure assessing how well it can discriminate traces at that node. For this parameter optimization problem, the *LoTuS* system uses Particle Swarm Optimization [43] and supports two types of the impurity measure: information gain and misclassification gain [44], [45].

To prevent the decision tree from over-fitting the training data, the algorithm does not recurse on a node if at least one of the following conditions is met.

- Tree depth at the node is 5.
- At least 98% of traces in the node belong to the same class.
- The node contains less than 5 traces.

On top of that, the post-completion cost-complexity pruning [46] is performed using a validation set.

The obtained decision tree can be directly used for classification of traces or converted to an equivalent STL formula. An example of such a tree and its corresponding STL formula is shown in Figure 1. Every path in the tree that leads to a leaf node with a positive class label corresponds to an STL formula obtained as a conjunction of formulae in the non-leaf nodes along this path such that the formula is negated if and only if its right subtree is followed next. The overall STL formula is formed as a disjunction of these conjunctions.

The *LoTuS* system supports both offline and online learning scenarios, and it was evaluated in two case studies: fault detection in an automotive system and anomaly detection in a maritime environment.

V. SOCIAL DINING DATASET

I conducted a controlled experiment to collect a novel dataset of audio-visual recordings of people eating together (Section V-A). I then extracted high-level features of participants' behaviors (Section V-B) and annotated them with ground-truths (Section V-C). Finally, I constructed the training examples (Section V-D).

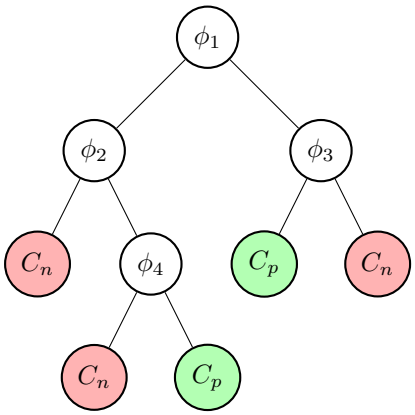

Fig. 1. Sample decision tree and its equivalent STL formula $\phi_{tree} = (\phi_1 \wedge (\neg\phi_2 \wedge \neg\phi_4)) \vee (\neg\phi_1 \wedge \phi_3)$. The positive class is denoted by C_p and the negative class by C_n.

A. Data collection

I recruited 24 people[1] (15 females and 9 males) among Cornell-affiliated fully-vaccinated students/faculty/staff to eat a meal together in triples for 30-45 minutes. 0 sessions were at breakfast time, 4 at lunch time, and 4 at dinner time. Each participant was 18+ years old and took part in the study only once. The study setup is illustrated in Figure 2. There are three cameras (mutually at 120°) in the middle of the table, each capturing one participant at a participant position $p \in \{1, 2, 3\}$. There was also a fourth camera capturing the whole scene which served only for documentation purposes. All four cameras were Intel RealSense Depth Camera D455 [47]. The scene audio was captured by a microphone array ReSpeaker Mic Array v2.0 [48] placed in the middle of the table. The ReSpeaker microphone array contains four microphones arranged at the corners of a square, estimates the direction of sound arrival, and performs voice activity detection. It also has LEDs that blink based on voice activity and direction. However, I turned them off in order not to distract participants.

Participants were free to bring any kind of food and utensils (most participants ate with a fork, spoon, or chopsticks and the rest with hands). They could also bring a drink (some brought cups, others bottles) and were provided with napkins (which most participants utilized). Before the study, each participant was asked to fill in a pre-study questionnaire about their demographic background, relationship to other participants, and social dining habits. They were then asked to eat their meals and have natural conversations. The instructor started the recording and left the room. When *all* three participants finished eating the recording was stopped and participants were asked to fill in a post-study questionnaire about their meal experience. The participants were compensated $15 for each half hour of participation and additional $15 as a meal reimbursement.

[1]For this project, I used the data only from one recording session (3 people) that I annotated so far.

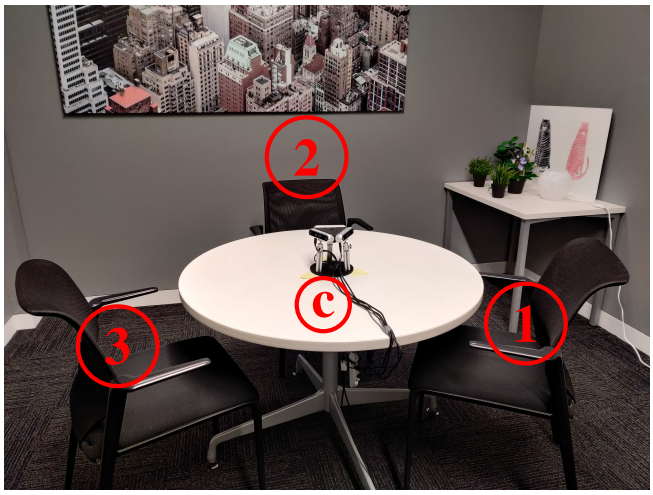

Fig. 2. Data collection setup. Participants at positions 1, 2, and 3 sit and eat at a round table. Each participant is recorded from the center (c) of the table by one Intel RealSense Depth Camera D455 [47]. The three cameras have mutual angles of 120°. The scene audio is captured by a microphone array ReSpeaker Mic Array v2.0 [48] located in the center (c) of the table.

B. Feature extraction

I chose head rotation angles as high-level features since I expect them to be informative for predicting bite transfer timings. In particular, they should be able to capture the individual participant's behavior such as looking down (e.g., while manipulating their food or thinking of taking a next bite) as well as the interactions between participants (e.g., looking at each other).

Using the *OpenFace* library [49] and a sampling rate of 30 Hz, for each participant i and for each time step t, I extracted the high-level feature vector

$$ {}_if_t = (\theta_p, \theta_y) $$

where θ_p and θ_y are the i-th participant's head rotation angles pitch and yaw, respectively, at the time step t. The feature vectors ${}_if_t$ correspond to samples from the continuous signal $f(t)$ defined in Section III. I further computed first-order differences of these angles as a feature vector ${}_if'_t = (d\theta_p, d\theta_y)$ which might be informative of the dynamics of movements, and the feature vector ${}_if''_t = (\theta_p, \theta_y, d\theta_p, d\theta_y)$ by concatenating ${}_if_t$ and ${}_if'_t$. I will further refer to feature vectors ${}_if_t$, ${}_if'_t$, and ${}_if''_t$ as feature types R_2, dR_2, and R_2dR_2 respectively.

For each participant i, the feature vectors ${}_if_t$ from a time window of w time steps starting at time t are then aggregated in the feature matrix

$$ {}_iF_{t:t+w} = \begin{bmatrix} {}_if_t \\ {}_if_{t+1} \\ \vdots \\ {}_if_{t+w-1} \end{bmatrix} $$

and analogously the feature matrices ${}_iF'_{t:t+w}$ and ${}_iF''_{t:t+w}$ for feature vectors ${}_if'_t$ and ${}_if''_t$ respectively. In general, each such feature matrix represents one *trace*.

C. Annotation

Using the ELAN annotation tool [50], I annotated each participant's video and assigned a `bite_to_mouth` annotation to all frames between the time when the food item entered the mouth and the time the utensil/hand left the mouth. To each trace[2] ${}_iF_{t:t+w}$, I then assigned a binary label ${}_iy_{t+w}$ corresponding to the participant i at time step $t+w$ such that

$$ {}_iy_{t+w} = \begin{cases} 1 & \text{if there is a } \texttt{bite\_to\_mouth} \text{ annotation at} \\ & \text{time step } t+w \\ 0 & \text{otherwise} \end{cases} $$

I refer to the label ${}_iy_{t+w} = 1$ as a positive label.

D. Training examples

Each study session provided three sets of data, each corresponding to one participant whose bite transfer timing I learned to predict. I will further refer to this participant as a *target participant*. On top of that, I created the following three kinds of training datasets that I will further refer to as *feature sources*.

- **Solo**: The training examples are of the form $({}_iF_{t:t+w}, {}_iy_{t+w})$ so that the traces of participant i's behavior can be used to predict the participant i's bite transfer timing.
- **Duo**: The training examples are of the form $(({}_jF_{t:t+w}, {}_kF_{t:t+w}), {}_iy_{t+w})$ so that the traces of participant j's and k's behaviors can be used to predict the participant i's bite transfer timing. The participants i, j, k belong to the same study session and $i \neq j$, $i \neq k$, and $j \neq k$.
- **Trio**: The training examples are of the form $(({}_iF_{t:t+w}, {}_jF_{t:t+w}, {}_kF_{t:t+w}), {}_iy_{t+w})$ so that the traces of all i, j, k participants' behaviors can be used to predict the participant i's bite transfer timing. The participants i, j, k belong to the same study session and $i \neq j$, $i \neq k$, and $j \neq k$.

For each kind of these datasets, I generated training examples from time step $t = 0$ up to $t = T$ where T is the time step when the last bite transfer for the participant i occurred, i.e., the largest T when ${}_iy_T = 1$. The reason is that after a participant finished eating, the use of subsequent annotations for this participant might be confounding for learning to predict appropriate bite transfer times.

The obtained datasets were highly imbalanced (3,095 positive examples and 91,312 negative examples each) and so I used the *imbalanced-learn* library [51] to randomly undersample the majority class to exactly match the number of examples in the minority class.

VI. METHODOLOGY

To learn the appropriate bite timings in terms of STL formulae, I used the STL learning system *LoTuS* [5], described in Section IV, with the original settings. As an

[2]In the rest of this section, I refer only to the trace ${}_iF_{t:t+w}$ of the feature type R_2 but the same applies to traces of feature types dR_2 and R_2dR_2.

impurity measure I chose the information gain as it tends to outperform the misclassification gain for deeper trees [46] and the authors of the *LoTuS* system already noticed this tendency for the maximum tree depth of 5.

I performed several experiments across the following hyperparameters and settings.

- Feature type $\Phi \in \{R_2, dR_2, R_2dR_2\}$.
- Window size $W \in \{15,\ 30,\ 45,\ 60,\ 75,\ 90,\ 105,\ 120,\ 135,\ 150,\ 165,\ 180,\ 195,\ 210\}$ corresponding to time intervals of 0.5-7 seconds.
- Feature source $S \in \{Solo, Duo, Trio\}$.
- Formulae primitives type $P \in \{P_1, P_2, P_{12}\}$ where I introduced a new set of primitives $P_{12} = P_1 \bigcup P_2$ as a combination of first-level and second-level primitives.

Using these experiments, I aimed to answer the following research questions.

1) *What is the optimal feature type and optimal window size that best capture the participant behavior relevant for predicting the bite transfer timing?*
   - For this experiment, I used the $Solo$ feature source and the P_1 primitives type.
2) *What combinations of feature type and primitives type result in the lowest misclassification rate (MCR) and how does it vary across various feature sources?*
   - For this experiment, I used the optimal window size found when answering the first research question.
3) *What combinations of feature type and primitives type achieve the lowest training time and how does it vary across various feature sources?*
   - For this experiment, I used the optimal window size found when answering the first research question.
4) *How can the obtained formulae be interpreted?*

Each experiment was evaluated using 5-fold cross-validation and I reported a mean and standard deviation of the test MCRs and training times over all folds.

VII. RESULTS AND DISCUSSION

The analysis of the obtained results for the four research questions outlined in Section VI is provided in the following four subsections respectively.

A. Optimal feature type and window size

As can be seen from Figure 3, for various feature types, there is no clear indication of the optimal window size other than that for the feature type $dR2$ it should be between 30 and 180 frames. To reduce the model complexity but still have a sufficient time window to capture behaviors relevant for bite timing prediction, I chose the optimal window size of $W = 75$ which corresponds to 2.5 seconds.

For various window sizes, the first-order differential features dR_2 on their own result in much higher MCR than the basic set of head rotation features R_2, however, their combination R_2dR2 achieves the lowest MCR. This suggests that it is best to use the feature type R_2dR2. However, this might vary for different feature sources and so I performed the following experiments using all feature types.

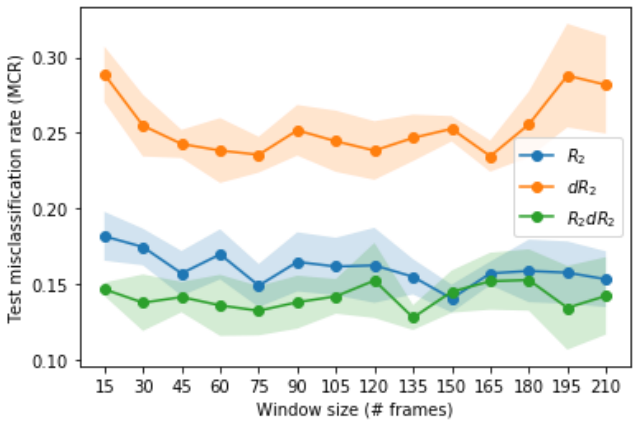

Fig. 3. Mean (and standard deviation) of test misclassification rate (MCR) from 5-fold cross-validation in terms of window size and for each feature type $\Phi \in \{R_2, dR_2, R_2dR_2\}$, using the $Solo$ feature source and the P_1 primitives type.

B. Misclassification rate (MCR) across feature types, primitives types, and feature sources

Table I shows that the optimal choice (Φ, P) of the feature type and primitives type for the $Solo$ feature source is (R_2dR2, P_1) and for the Duo and $Trio$ feature sources it is (R_2dR2, P_{12}). This suggests that there is no specific optimal feature type for each feature source, and so the finding (that it is best to use the feature type R_2dR2) from the previous subsection can be generalized to feature sources Duo and $Trio$. It may further indicate that the more complex interactions in case of the feature sources Duo and $Trio$ may require bigger variety of formulae primitives offered by the primitives set P_{12}.

Comparing between the primitives types in general, the primitives types P_1 and P_{12} tend to achieve lower MCR than P_2 but there do not seem to be considerable differences. So there might not be a significant benefit from using higher-order primitives.

Comparing between the feature sources in general, the Duo feature source achieves around twice the MCR of the other feature sources which suggests that the features from the target participant are much more informative for prediction of bite transfer timings than the features from the other two participants.

C. Decision tree training time across feature types, primitives types, and feature sources

As shown by Table II, for any feature source and any feature type, the decision tree training time is considerably shorter (about 20-times) for the primitives type P_1 than for the primitives types P_2 and P_{12}, while it is very similar between the primitives types P_2 and P_{12}. Together with the findings from the previous subsection this implies that the potentially slightly lower MCR achieved with the primitives

TABLE I

MEAN (AND STANDARD DEVIATION) OF TEST MISCLASSIFICATION RATE (MCR) FROM 5-FOLD CROSS-VALIDATION. CELL COLORS CORRESPOND TO MEAN TEST MCR ON A SCALE FROM 0.114 (YELLOW) TO 0.412 (RED).

		Primitives type		
Feature source	Feature type	P_1	P_2	P_{12}
$Solo$	R_2	0.149 (0.014)	0.164 (0.004)	0.162 (0.019)
	dR_2	0.236 (0.012)	0.250 (0.016)	0.202 (0.010)
	R_2dR_2	0.132 (0.016)	0.146 (0.026)	0.135 (0.013)
Duo	R_2	0.308 (0.026)	0.340 (0.016)	0.314 (0.008)
	dR_2	0.412 (0.024)	0.412 (0.027)	0.396 (0.018)
	R_2dR_2	0.316 (0.009)	0.315 (0.021)	0.300 (0.007)
$Trio$	R_2	0.143 (0.014)	0.164 (0.013)	0.137 (0.009)
	dR_2	0.224 (0.015)	0.227 (0.023)	0.214 (0.016)
	R_2dR_2	0.144 (0.013)	0.152 (0.024)	0.114 (0.006)

type P_{12} might not be worth the significantly higher training time as compared to the primitives type P_1.

D. Formulae interpretation

In Table III, I present sample inferred STL formulae learned using the feature type R_2 and the primitives types P_1 for various feature sources $S \in \{Solo, Duo, Trio\}$.

For example, the formula ϕ_{solo} captures the rule (highlighted in blue) that *in the time interval 1.21-2.46 seconds the pitch angle has to be less than 0.362 rad* which seems reasonable that the participant looks down at their food before they take a bite at a time step of 2.5 seconds. However, it also captures several rules such as *the yaw angle is less than -0.447 rad within 1.7-2.47 seconds* (highlighted in red) which suggests that the person is looking sideways just before taking a bite which does not make much sense and might be the result of overfitting. Also, if the start and end times of a time interval are very close to each other it may indicate overfitting (highlighted in red).

Overall, I can conclude that stronger regularization during training such as restricting the maxium tree depth to 3 might be beneficial and can result in more generalizable formulae. Therefore, I also experimented with the maximum tree depth of 3 in which case I chose the optimal window size $W = 60$ that corresponds to 2 seconds. Sample decision trees corresponding to inferred STL formulae learned using the feature source Duo, feature type R_2, and primitives type P_1 are shown in Figure 4. The possible interpretations of the rules for the appropriate bite transfer timing are as follows.

- *The topmost tree:* When the participant k eventually looks at the other non-target participant j within 1.36-1.77 seconds.
- *The middle tree:* When the participant k eventually looks at their food (or possibly at the other non-target participant j) within 1.48-1.86 seconds and if the participant j eventually looks down at their food within 0.02-0.27 seconds.
- *The bottom tree:* When the participant k always looks at their food (or possibly at the other non-target participant j) within 1.39-1.93 seconds. Or when the participant j eventually looks at the participant k within 0.50-1.12 seconds and the participant k is not eating within 0.00-0.55 seconds.

VIII. CONCLUSIONS

This work addressed the problem of modeling human eating behaviors in terms of Signal Temporal Logic (STL) formulae. Specifically, the goal was to predict the ideal bite transfer timing in a social dining setting which can be then used to drive an assistive robot that feeds a person with disabilities.

I first collected a novel audio-visual dataset capturing triadic interactions of people eating together and then used the *LoTuS* system [5] to learn the STL formulae that decide the appropriate bite transfer timings. I experimented with various window sizes, feature types, feature sources (from the target participant only, from the other two participants, or from all three participants), and formulae primitives types.

The results show that: 1) the most informative features is the combination of head rotation angles and their first-order differences; 2) there does not seem to be one specific length of the time interval that is informative for bite transfer timing prediction; 3) the features from the target participant are much more important for the prediction of bite transfer timings for that participant than the features from the other two participants; and 4) the first-order formulae primitives might be sufficient and allow much faster training.

IX. FUTURE WORK

The following ideas can be investigated as future work.

- Add the time t_{slb} since the last bite transfer as another feature to the feature vector.
- Add more features (such as hand and mouth positions and voice activity) that can be extracted from audio-visual recordings using *OpenPose* [52] for human pose tracking and *py-webrtcvad* [53] for voice activity detection.
- In the case of the $Trio$ feature source, try different sets of features for the target participant and the other two participants since there might not be one set of features that is optimal for both the target participant and the other two participants.

TABLE II

MEAN (AND STANDARD DEVIATION) OF DECISION TREE TRAINING TIME IN SECONDS FROM 5-FOLD CROSS-VALIDATION. CELL COLORS CORRESPOND TO MEAN DECISION TREE TRAINING TIME ON A SCALE FROM 16 SECONDS (YELLOW) TO 2034 SECONDS (RED).

		Primitives type		
Feature source	Feature type	P_1	P_2	P_{12}
Solo	R_2	16 (1)	449 (18)	433 (22)
	dR_2	18 (2)	368 (52)	405 (24)
	R_2dR_2	34 (2)	827 (46)	772 (37)
Duo	R_2	30 (2)	702 (30)	729 (47)
	dR_2	26 (1)	473 (52)	504 (49)
	R_2dR_2	70 (3)	1218 (67)	1282 (84)
Trio	R_2	48 (2)	1007 (71)	1033 (45)
	dR_2	48 (4)	951 (28)	943 (13)
	R_2dR_2	136 (19)	1858 (75)	2034 (138)

TABLE III

SAMPLE INFERRED STL FORMULAE LEARNED USING THE MAXIMUM DECISION TREE DEPTH OF 5, FEATURE TYPE R_2 AND THE PRIMITIVES TYPE P_1 FOR VARIOUS FEATURE SOURCES. THE SUBSCRIPT i DENOTES THE FEATURES OF THE TARGET PARTICIPANT WHILE j AND k DENOTE THE FEATURES OF THE OTHER TWO PARTICIPANTS.

Feature source	Formula
Solo	$\phi_{Solo} = (\square_{[1.21,2.46]\,i}\theta_p < 0.362 \wedge (\diamond_{[0.136,1.79]\,i}\theta_p > 0.11 \wedge (\square_{[1.7,2.47]\,i}\theta_y < -0.447 \vee (\diamond_{[1.7,2.47]\,i}\theta_y > -0.447 \wedge (\diamond_{[1.78,2.46]\,i}\theta_p > 0.119 \wedge \diamond_{[1.46,2.47]\,i}\theta_y > 0.561))))) \vee (\diamond_{[1.21,2.46]\,i}\theta_p > 0.362 \wedge ((\square_{[2.4,2.47]\,i}\theta_y < 0.171 \wedge (\diamond_{[2.01,2.43]\,i}\theta_y > -0.0336 \wedge \diamond_{[0.674,1.27]\,i}\theta_y < 0.0813)) \vee \diamond_{[2.4,2.47]\,i}\theta_y > 0.171))$
Duo	$\phi_{Duo} = ((\square_{[1e-06,2.43]\,k}\theta_y > 0.0272 \wedge (\diamond_{[0.0117,2.16]\,k}\theta_p > 0.474 \wedge (\diamond_{[0.0788,0.56]\,j}\theta_p > 0.0613 \wedge \square_{[1e-06,2.41]\,k}\theta_y < 0.536))) \vee (\diamond_{[1e-06,2.43]\,k}\theta_y < 0.0272 \wedge ((\square_{[0.299,2.18]\,k}\theta_y > -0.704 \wedge (\diamond_{[0.176,2.4]\,j}\theta_y > -0.0922 \wedge (\square_{[0.126,0.167]\,j}\theta_p < 0.315 \vee (\diamond_{[0.126,0.167]\,j}\theta_p > 0.315 \wedge \square_{[1.58,2.38]\,j}\theta_y > 0.153)))) \vee \diamond_{[0.299,2.18]\,k}\theta_y < -0.704)))$
Trio	$\phi_{Trio} = ((\square_{[1.26,2.47]\,i}\theta_p < 0.361 \wedge (\diamond_{[1e-06,1.8]\,i}\theta_p > 0.111 \wedge (\square_{[1e-06,2.46]\,j}\theta_y > 0.188 \wedge (\square_{[0.641,2.25]\,j}\theta_p > 0.0174 \wedge \square_{[0.0275,2]\,i}\theta_y < 0.161)))) \vee (\diamond_{[1.26,2.47]\,i}\theta_p > 0.361 \wedge (\square_{[2.45,2.47]\,i}\theta_y > 0.159 \vee (\diamond_{[2.45,2.47]\,i}\theta_y < 0.159 \wedge (\diamond_{[2.02,2.47]\,i}\theta_y > -0.0528 \wedge (\diamond_{[1e-06,1.58]\,i}\theta_y < 0.0792 \wedge \diamond_{[0.0733,1.96]\,j}\theta_y > 0.193))))))$

- Experiment with a suitable time delay between the end of the window of features and the predicted label. This might help eliminate confounding information captured immediately before the bite transfer and which might not be relevant for the bite transfer prediction.
- Compare with other methods such as the systematic enumeration of STL formulae with heuristic pruning by Mohammadinejad et al. [11] and their follow-up work [54] that uses counter-example guided inductive synthesis.
- Experiment with the data uncertainty-aware approach to STL inference [55].
- Use boosted decision trees for STL formulae learning, similarly to the works [16], [17]. Also, experiment with various regularization techniques for decision tree learning.
- Compare this rule-based approach to a purely data-driven prediction of the ideal bite timing using methods such as Temporal Convolutional Networks [56] or PazNet [57].
- Perform qualitative evaluation of the inferred formulae. Rephrase the formulae in natural language and ask people to rate (for example, via a web survey using a Likert scale 1-5) how realistic the learned rule is.
- Learn STL formulae for a specific person and compare them with the formulae learned in a person-independent manner, investigating intra-person and inter-person variability.
- Perform experiments in the online learning setting, similarly to the work [5].
- Perform significance testing for the various cross-validation experiments to see if the differences are significant.

REFERENCES

[1] World Health Organization and The World Bank. World report on disability. https://www.who.int/disabilities/world_report/2011/report.pdf, 2011.

[2] Lin Perry. Assisted feeding, 2008.

[3] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[4] Rolf Isermann. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2005.

[5] Giuseppe Bombara and Calin Belta. Offline and online learning of signal temporal logic formulae using decision trees. *ACM Transactions on Cyber-Physical Systems*, 5(3):1–23, 2021.

[6] Fujun Wang, Zining Cao, Lixing Tan, and Hui Zong. Survey on learning-based formal methods: Taxonomy, applications and possible future directions. *IEEE Access*, 8:108561–108578, 2020.

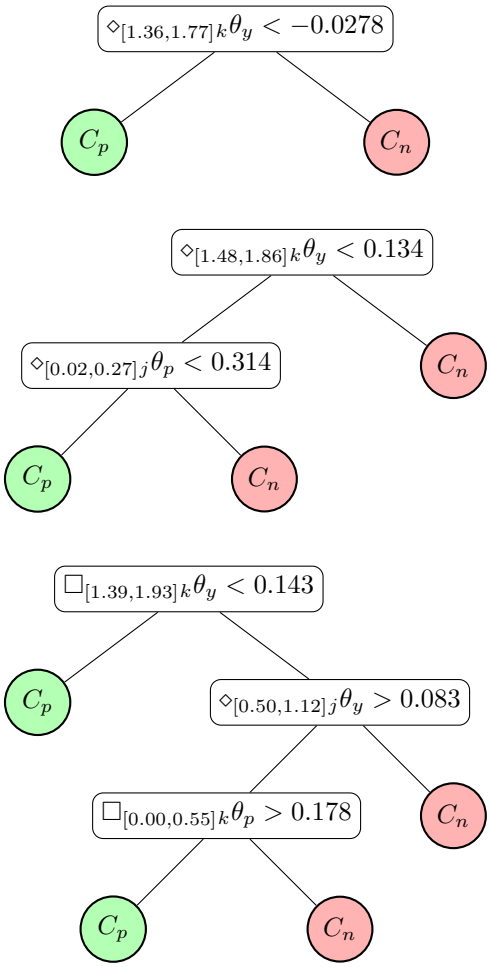

Fig. 4. Sample decision trees corresponding to inferred STL formulae learned using the maximum decision tree depth of 3, window size 60, feature source Duo, feature type R_2, and the primitives type P_1. The subscripts j and k denote the features of the other two participants, i.e., not the target participant. The positive class is denoted by C_p and the negative class by C_n.

[7] Zhaodan Kong, Austin Jones, and Calin Belta. Temporal Logics for Learning and Detection of Anomalous Behavior. *IEEE Trans. Automat. Contr.*, 62(3):1210–1222, March 2017.

[8] Ezio Bartocci, Luca Bortolussi, and Guido Sanguinetti. Data-driven statistical learning of temporal logic properties. In *International conference on formal modeling and analysis of timed systems*, pages 23–37. Springer, 2014.

[9] Sara Bufo, Ezio Bartocci, Guido Sanguinetti, Massimo Borelli, Umberto Lucangelo, and Luca Bortolussi. Temporal logic based monitoring of assisted ventilation in intensive care patients. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, pages 391–403. Springer, 2014.

[10] Laura Nenzi, Simone Silvetti, Ezio Bartocci, and Luca Bortolussi. A robust genetic algorithm for learning temporal specifications from data. In *International Conference on Quantitative Evaluation of Systems*, pages 323–338. Springer, 2018.

[11] Sara Mohammadinejad, Jyotirmoy V Deshmukh, Aniruddh G Puranic, Marcell Vazquez-Chanlatte, and Alexandre Donzé. Interpretable classification of time-series data using efficient enumerative techniques. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–10, 2020.

[12] Giuseppe Bombara, Cristian-Ioan Vasile, Francisco Penedo, Hirotoshi Yasuoka, and Calin Belta. A decision tree approach to data classification using signal temporal logic. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 1–10, 2016.

[13] Zhaodan Kong, Austin Jones, Ana Medina Ayala, Ebru Aydin Gol, and Calin Belta. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 273–282, 2014.

[14] Chanyeol Yoo and Calin Belta. Rich time series classification using temporal logic. In *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation, July 2017.

[15] Mingguang Hu and Zining Cao. Lrx: Specification mining based on logistic regression. In *Proceedings of the 2020 4th International Conference on Management Engineering, Software Engineering and Service Sciences*, pages 68–72, 2020.

[16] Erfan Aasi, Cristian Ioan Vasile, Mahroo Bahreinian, and Calin Belta. Classification of time-series data using boosted decision trees. *arXiv preprint arXiv:2110.00581*, 2021.

[17] Erfan Aasi, Cristian Ioan Vasile, Mahroo Bahreinian, and Calin Belta. Inferring temporal logic properties from data using boosted decision trees. *arXiv preprint arXiv:2105.11508*, 2021.

[18] Susmit Jha, Ashish Tiwari, Sanjit A Seshia, Tuhin Sahai, and Natarajan Shankar. Telex: Passive stl learning using only positive examples. In *International Conference on Runtime Verification*, pages 208–224. Springer, 2017.

[19] Susmit Jha, Ashish Tiwari, Sanjit A Seshia, Tuhin Sahai, and Natarajan Shankar. TeLEx: learning signal temporal logic from positive examples using tightness metric. *Formal Methods in System Design*, 54(3):364–387, November 2019.

[20] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In *International Conference on Runtime Verification*, pages 147–160. Springer, 2011.

[21] Alexey Bakhirkin, Thomas Ferrere, and Oded Maler. Efficient parametric identification for stl. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 177–186, 2018.

[22] Hengyi Yang, Bardh Hoxha, and Georgios Fainekos. Querying parametric temporal logic properties on embedded systems. In *IFIP International Conference on Testing Software and Systems*, pages 136–151. Springer, 2012.

[23] François Fages and Aurélien Rizk. From model-checking to temporal logic constraint solving. In *International Conference on Principles and Practice of Constraint Programming*, pages 319–334. Springer, 2009.

[24] Amir Pnueli. The temporal semantics of concurrent programs. *Theoretical computer science*, 13(1):45–60, 1981.

[25] Rüdiger Ehlers, Ivan Gavran, and Daniel Neider. Learning Properties in LTL ∩ ACTL from Positive Examples Only. *2020 Formal Methods in Computer Aided Design (FMCAD)*, pages 104–112, 2020.

[26] Ezio Bartocci, Luca Bortolussi, Laura Nenzi, and Guido Sanguinetti. System design of stochastic models using robustness of temporal properties. *Theoretical Computer Science*, 587:3–25, 2015.

[27] Gang Chen, Zachary Sabato, and Zhaodan Kong. Active learning based requirement mining for cyber-physical systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4586–4593, 2016.

[28] Bardh Hoxha, Adel Dokhanchi, and Georgios Fainekos. Mining parametric temporal logic properties in model-based design for cyber-physical systems. *International Journal on Software Tools for Technology Transfer*, 20(1):79–93, 2018.

[29] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy V. Deshmukh, and Sanjit A. Seshia. Mining requirements from closed-loop control models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(11):1704–1717, 2015.

[30] Marcell Vazquez-Chanlatte, Jyotirmoy V Deshmukh, Xiaoqing Jin, and Sanjit A Seshia. Logical clustering and learning for time-series data. In *International Conference on Computer Aided Verification*, pages 305–325. Springer, 2017.

[31] Marcell Vazquez-Chanlatte, Shromona Ghosh, Jyotirmoy V Deshmukh, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Time-series learning using monotonic logical properties. In *International Conference on Runtime Verification*, pages 389–405. Springer, 2018.

[32] Austin Jones, Zhaodan Kong, and Calin Belta. Anomaly detection in cyber-physical systems: A formal methods approach. In *53rd IEEE Conference on Decision and Control*, pages 848–853. IEEE, 2014.

[33] Giuseppe Bombara and Calin Belta. Online learning of temporal logic formulae for signal classification. In *2018 European Control Conference (ECC)*, pages 2057–2062. IEEE, 2018.

[34] Alexis Linard and Jana Tumova. Active learning of signal temporal logic specifications. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 779–785. IEEE, 2020.

[35] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[36] Szymon Stoma, Alexandre Donzé, François Bertaux, Oded Maler, and Gregory Batt. Stl-based analysis of trail-induced apoptosis challenges the notion of type i/type ii cell line classification. *PLoS computational biology*, 9(5):e1003056, 2013.

[37] Zhe Xu, Calin Belta, and Agung Julius. Temporal logic inference with prior information: An application to robot arm movements. *IFAC-PapersOnLine*, 48(27):141–146, 2015.

[38] Oded Maler and Dejan Ničković. Monitoring properties of analog and mixed-signal circuits. *International Journal on Software Tools for Technology Transfer*, 15(3):247–268, 2013.

[39] James Kapinski, Xiaoqing Jin, Jyotirmoy Deshmukh, Alexandre Donze, Tomoya Yamaguchi, Hisahiro Ito, Tomoyuki Kaga, Shunsuke Kobuna, and Sanjit Seshia. St-lib: A library for specifying and classifying model behaviors. Technical report, SAE Technical Paper, 2016.

[40] Xiaoqing Jin, Jyotirmoy V Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. Powertrain control verification benchmark. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 253–262, 2014.

[41] Ayca Balkan, Paulo Tabuada, Jyotirmoy V Deshmukh, Xiaoqing Jin, and James Kapinski. Underminer: A framework for automatically identifying nonconverging behaviors in black-box system models. *ACM Transactions on Embedded Computing Systems (TECS)*, 17(1):1–28, 2017.

[42] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

[43] Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, pages 69–73. IEEE, 1998.

[44] Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[55] Nasim Baharisangari, Jean-Raphaël Gaglione, Daniel Neider, Ufuk Topcu, and Zhe Xu. Uncertainty-aware signal temporal logic. *arXiv preprint arXiv:2105.11545*, 2021.

[45] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[46] Jerome H Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017.

[47] Introducing the Intel® realsense™ depth camera D455. `https://www.intelrealsense.com/depth-camera-d455/`, Nov 2021.

[48] Baozhu Zuo. ReSpeaker Mic Array v2.0. `https://wiki.seeedstudio.com/ReSpeaker_Mic_Array_v2.0/`.

[49] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 59–66. IEEE, 2018.

[50] Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann, and Han Sloetjes. Elan: A professional framework for multimodality research. In *5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 1556–1559, 2006.

[51] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[52] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[53] py-webrtcvad: Python Interface to the WebRTC Voice Activity Detector. `https://github.com/wiseman/py-webrtcvad/`.

[54] Sara Mohammadinejad, Jyotirmoy V Deshmukh, and Aniruddh G Puranic. Mining environment assumptions for cyber-physical system models. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, pages 87–97. IEEE, 2020.

[56] Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision*, pages 47–54. Springer, 2016.

[57] Tong Wu, Nikolas Martelaro, Simon Stent, Jorge Ortiz, and Wendy Ju. Learning when agents can talk to drivers using the inagt dataset and multisensor fusion. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(3), September 2021.