A Framework for the Unsupervised Inference of Relations Between Sensed Object Spatial Distributions and Robot Behaviors

Christopher Morse[1], Lu Feng[1], Matthew Dwyer[1], and Sebastian Elbaum[1]

***Abstract*— The spatial distribution of sensed objects strongly influences the behavior of mobile robots. Yet, as robots evolve in complexity to operate in increasingly rich environments, it becomes much more difficult to exhaustively specify the underlying relations between spatial object distributions and robot behavior. We aim to address this challenge by leveraging system trace data to automatically infer relations that help to better characterize these spatial associations. In particular, we introduce SpRInG, a framework for the unsupervised inference of system specifications that characterize the spatial relationships under which a robot operates. Our method builds on a parameterizable notion of reachability to encode relationships of spatial neighborship, which are used to instantiate a language of patterns. These patterns then provide the structure to infer from the traces the connection between such relationships and robot behaviors. We show that SpRInG can automatically infer spatial relations on two distinct domains: autonomous vehicles in traffic and a teleoperated surgical robot. Our results demonstrate the power and expressiveness of SpRInG, in its ability to learn existing system specifications as machine-checkable first-order logic, uncover previously unstated system specifications that are rich and insightful, and reveal contextual differences between executions.**

I. INTRODUCTION

Robot behavior is often guided by the spatial distribution of sensed objects in its environment. For example, consider a scenario in which autonomous vehicles (AVs) are tasked to navigate through highway traffic (see Figure 1). In this scenario, we expect the AV in the top lane to change lanes only if it senses that there are no vehicles blocking its target lane. Furthermore, we may observe that the same AV will decelerate to allow its nearby vehicle to pass.

As robot systems and their operating environments grow in complexity, the underlying relation between the spatial distribution of sensed objects and robot behavior becomes increasingly difficult to specify. This challenge is compounded by several factors. First, there is a high dimensional input space, yielding an extensively large quantity of potential distributions of objects. In the traffic scenario, each AV contains a wide array of state variables (e.g. *pose, velocities, sensor readings*) and potential values; manually defining relevant relationships between all entities and variables quickly becomes infeasible. Second, the increasing number of learned components in robots tends to obscure the relationship between sensed object distributions and robot behaviors. In the traffic scenario, the behavior of each AV may not be explicitly stated in the code, but rather governed

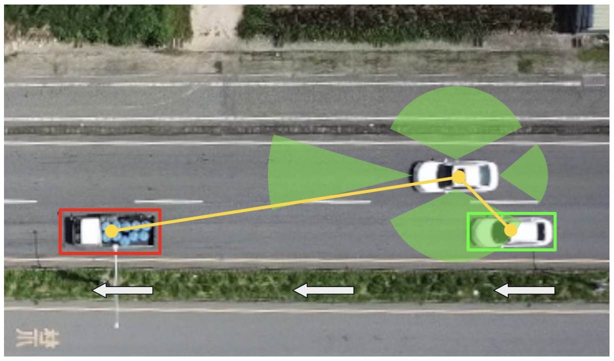

Fig. 1. Autonomous vehicles operating in a traffic scenario. The boxed vehicles (bottom lane) are those that are detected by the AV in the top lane, and the yellow lines correspond to their sensed distances. If a vehicle is detected within one of the shaded green regions around the AV, then it is determined to have direct influence over the AV's behavior (green box) or negligible influence (red box). The white arrows indicate the direction of traffic.

by a black-box DNN that accepts a LiDAR reading as input, detects relevant objects, and outputs control signals. For such a model, the decision-making process is not always evident or explainable. Third, it is infeasible to anticipate how a robot will handle high unpredictability in complex scenarios. In the traffic scenario, several factors lead to uncertainty. These include the variability of the operating environment, the heterogeneity of sensor capabilities and equipment, and, with a human in the loop, the unpredictability of human behavior.

Each of these complexities greatly exacerbates the cost of specifying the behavior for robot systems, which is why many remain unspecified. Because the misbehavior of robots can lead to catastrophic outcomes, it is important to define these specifications and safely abort upon violation. To address this challenge, we aim to automatically infer relations that help to characterize the relationships between sensed object spatial distributions and robot behaviors, by leveraging the increasingly rich sensor data collected by deployed robots. In the context of the traffic scenario, we aim to capture relations like:

- *If an AV senses that its lead vehicle decelerates, then the AV will* ***decelerate****.*
- *If an AV senses a nearby vehicle in its left lane, then the AV would* ***not perform a left lane change****.*
- *When an AV senses a red traffic light ahead, then the AV will* ***stop****.*

Such inferred relations have several desirable attributes. First, they possess a level of abstraction that facilitates the connection between object spatial distributions and robot

[1]University of Virginia, USA {*drb6yv, lu.feng, matthewbdwyer, selbaum*}@*virginia.edu*

This work was funded in part by NSF Awards #1924777 and AFOSR #FA9550-21-1-0164.

behaviors. Second, they are inferred with no prior knowledge of system specifications, as they are learned solely from generated traces. Third, they are distinct from the specifications embedded in the robots' implementation, in that they include non-obvious relations between the distribution of its sensed objects and decisions made by its learned components.

To automatically infer such spatial relations, we introduce the *Spatial Relation Inference Generator* (SpRInG). This framework abstracts a trace of logged data into a sequence of graphs that encode parameterizable notions of reachability, prior to building on a rich language of patterns to instantiate spatial relations. While existing approaches either lack the infrastructure to learn spatial parameters [1] or have limited applicability for the inference of rich spatial relations from complex robot systems [2], SpRInG generates spatial relations that consider the inherent complexities and restrictions underlying such systems.

The primary contributions of this work are:

- A novel framework (SpRInG) for characterizing the spatial relationships between sensed object distributions and robot behaviors.
- An assessment of SpRInG over two distinct domains, to demonstrate its ability to automatically generate insightful spatial relations over diverse systems.

II. Related Work

The work on automated inferences from variable-value traces can be organized along three dimensions.

First, by the types of relations inferred and the logic by which they are encoded. Some approaches, such as Daikon [3], infer stateless linear relations in first-order logic that reflect the state of a system at a single point in time. Others, such as DIG [4], can infer stateless non-linear properties. More sophisticated methods can infer stateful temporal properties that account for the ordering of events, through the use of simple pattern matching [5], or genetic algorithms to infer signal temporal logic (STL) parameters [6].

Second, by the information that the user must provide to the inference process. For example, while Daikon and DIG can operate directly on traces by instantiating their entire set of predefined patterns, they are more efficient if the user provides a set of target trace variables. In approaches that infer temporal relations, the user may be required to bound the analysis by providing their own set of (at times, partially-instantiated) patterns [7, 8, 1].

Third, by the way in which inferred relations are used. Software developers have used inferred relations to serve as oracles for the testing process [9, 10], while such relations have also been used for verification [11], debugging [12], robot system monitoring [13], predictive monitoring [14, 15, 16], and to characterize uncertainty in robot systems [14].

Despite this body of research, there is limited work on characterizing the relationship between sensed object spatial distributions and robot behaviors. There have been efforts to specify and monitor spatial aspects of physical systems with spatial logics (e.g. SSTL [17], STREL [18], SpaTeL [19], and SaSTL [20]), but these approaches do not perform inference. The only existing work that makes inferences over spatial-temporal data constructs a spatial model as a graph, and aims to infer time and distance parameters for PSTREL formulae that are satisfied by all relevant nodes [2]. While this approach demonstrates its applicability for distributed stationary systems, the extension of their approach to mobile robots is limited. First, their inferred relations are limited by the inexpressiveness of STREL, which has been shown to only capture a small subset of complex specifications, relative to modern spatial logics [20]. Even further, our approach can extract much richer spatial relationships than distance bounds on spatial operators. Second, their spatial models preserve connectivity between nodes, an assumption that does not hold for mobile robots. Third, their approach requires the user to have substantial domain knowledge to provide patterns and parameter ordering. In contrast, our approach automatically provides an extensible library of patterns, thereby balancing automated discovery with pattern-query specificity.

III. Approach

The goal of SpRInG is to leverage rich system traces to automatically learn the underlying relations between the spatial distribution of sensed objects and the resulting robot behavior. The framework consists of three main components (see Figure 2). The *Spatial Encoder* component converts a raw trace into a sequence of graphs, which encode relevant spatial information. The *Inference Engine* component then infers spatial relations over these graphs, in the form of predicates, implications, and generalizations. These inferred relations then pass through a *Filter* prior to reporting. [1]

A. Spatial Encoder

The objective of the Spatial Encoder is to convert the trace into a sequence of graphs that succinctly encode relevant spatial information and align with existing spatial logics [17, 18, 20] (see Figure 2: "Spatial Encoder"). A *trace* contains a time-ordered sequence of *observations*, where each observation holds state and perception information for each entity in the system (e.g. drone, autonomous car, person). This information is stored as a set of variable-value pairs. The conversion of trace T into a sequence of graphs T_r is accomplished through its two subcomponents: the *Spatial Model Constructor* and the *Reachability Encoder*.

1) Spatial Model Constructor: The purpose of the *Spatial Model Constructor* is to initialize all possible spatial relationships between entities, which will later be used to discern and encode only those that are deemed the most valuable.

Each graph g_i is instantiated from an observation o_i by abstracting each system entity as a node $n_j \in V = \{n_1, n_2, ...n_n\}$. Each node is assigned information from o_i regarding its identifying information and state $s_{i,j}$ (e.g. ID, class, velocities, battery level). A directed edge is then constructed between each pair of entities (n_j, n_k), as in $e_{jk} \in E = \{e_{12}, e_{21}, ...e_{mn}\}$. Each edge e_{jk} contains a set

[1]Further information regarding our implementation is provided in our GitHub repository: https://github.com/less-lab-uva/SpRInG.

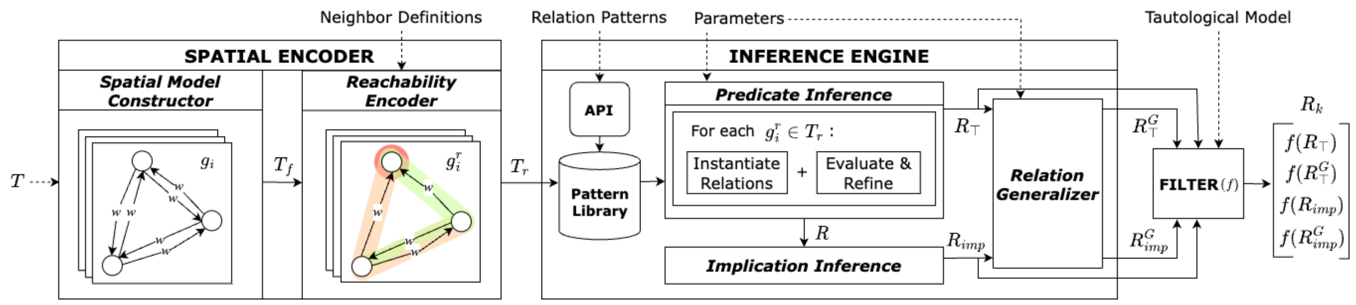

Fig. 2. The SpRInG framework. Its three main components are shown: *Spatial Encoder, Inference Engine*, and the *Filter*. In the *Reachability Encoder* subcomponent, neighbor relationships between entities are visualized with highlighted edges.

of attributes, $w_{jk} \in W$, that holds information to relate the paired entities. These entities are related by their available state information (from the intersection of both entities' state variable sets, i.e. $s_{i,j} \cap s_{i,k}$) and relevant perception information from entity n_j. After iterating this process for all observations in trace T, each $o_i \in T$ has been abstracted into its associated fully-connected graph, $g_i \in T_r$.

2) Reachability Encoder: The objective of the *Reachability Encoder* is to prune irrelevant relationships established in each fully-connected graph, such that the *Inference Engine* only learns relations over the most meaningful relationships. In particular, this subcomponent encodes the *reachability* between entities, a special form of spatial relationship in which one entity can be influenced by another (directly or indirectly). The conditions that must be met for one entity's behavior to be directly influenced by another are provided through a logical formula called a *neighbor definition*.

Definition 1 (**Neighbor Definition**). Given an entity $n_j \in \{n_1, n_2, ...n_n\}$, its neighbor definition $\Phi_j \in \Phi = \{\Phi_1, \Phi_2, ...\Phi_n\}$ is a propositional formula that states the requirements that must be satisfied by edge attributes from w_{jk}, for the behavior of n_j to be directly influenced by any other entity n_k. Due to the heterogeneity of sensing equipment and their limitations (e.g. range, noise), each entity is assigned its own neighbor definition. To evaluate such a formula, this component considers the following mapping: $z : (\Phi \times 2^W) \rightarrow True|False$, where W is the set of all edge attributes from some graph $g_i \in T_f$. This mapping is used to determine whether a set of edge attributes $w_{jk} \subseteq W$ satisfies Φ_j. After retrieving relevant values from w_{jk}, the entire neighbor definition is $True$ (edge is preserved) or $False$ (edge is removed). More generally, each entity can have multiple neighbor definitions to accommodate different *types* of neighbors (e.g. one for *leftNeighbors*, another for *rightNeighbors*).

Example 1. In the traffic scenario, assume that the user provides the following neighbor definitions as part of the input: $\Phi_{car_1} = (distance < 10) \wedge (detect\_confidence > 0.8)$ This states that the edge between car_1 and another entity car_k will only be preserved if car_k is less than 10 meters away and is detected with over 80% confidence.

The preserved relationships between pairs of entities form the basis for the construction of *reachability graphs*, where each graph encodes two abstract relationships for inference: *neighbors* and *neighborhoods*.

Definition 2 (**Reachability Graph**). A *reachability graph* $g_i^r \in T_r$ is a subgraph of $g_i = (V, E, W)$, denoted by $g_i^r = (V', E', W')$ with $V' = V$, $E' \subseteq E$, and $W' \subseteq W$, where the set of attributes $w'_{jk} \subseteq W'$ for each edge $n_j \rightarrow n_k = e'_{jk} \in E'$ satisfies $z(\Phi_j, w'_{jk}) = True$.

Definition 3 (**Neighborship**). Let $g_i^r \in T_r$ be a reachability graph. The *neighbors* of some node n_j at timestep i is the set of all entities that are adjacent to n_j in g_i^r (i.e. "directly reachable"). Intuitively, this is equivalent to the set of entities that do not violate Φ_j) at timestep i. The *neighborhood* of some node n_j at timestep i is the set of all entities that are returned from a graph search starting from n_j (i.e. "directly or indirectly reachable"), in the reachability graph g_i^r. Each of these are optionally predicated under the *type* of neighbor (see Section III-C).

B. Inference Engine

The goal of the inference engine component (see Figure 2: "Inference Engine") is to learn spatial relations from the trace of reachability graphs. Since these graphs encode the *reachability* between entities through *neighborship*, the learned relations are based on the *neighbor* and *neighborhood* relationships.

1) Pattern Library: The objective of the *Pattern Library* is to automatically generate and compile relation patterns such that they can be instantiated, evaluated, and refined by the *Predicate Inference* subcomponent. While a library of patterns can be automatically constructed with entity data from T_r, the user may choose to connect through an API to provide their own patterns. In either case, each relation must abide by a grammar. Further implementation details and the full grammar are provided in our GitHub repository.

Example 2. Consider a subset of the full grammar:

Productions	**Supporting Definitions**
$P_{main} \rightarrow P_1 \mid P_2 \mid P_3$	
$P_1 \rightarrow P_{main}$ OP P_{main}	OP: $\geq \mid \leq \mid == \mid \subset \mid \supset \mid \Rightarrow \mid \in \mid \notin$
$P_2 \rightarrow N$.Relation.Attribute	Relation: *Neighbors* $\mid$ *Neighborhood*
$P_3 \rightarrow CONST$	Attribute: *Size*

The relation pattern $(N.Neighbors.size \geq CONST)$ is constructed by $(P_{main}) \rightarrow (P_1) \rightarrow (P_{main}\ OP\ P_{main}) \rightarrow$

$(P_2\ OP\ P_3) \rightarrow (N.Relation.Attribute \geq CONST) \rightarrow (N.Neighbors.Size \geq CONST)$. Each $(a) \rightarrow (b)$ signifies that symbol (b) is a direct derivation from symbol (a), per the production rules. Each pattern must contain viable tokens (e.g. $CONST$ and N), which are filled during inference.

2) Predicate Inference: The objective of the *Predicate Inference* subcomponent is to use the relation patterns, along with the trace of reachability graphs T_r and a set of inference parameters for engine configuration, to produce a set of relations, R, that are either reported as stand-alone relations, $R_\top \subseteq R$, or used as predicates by the subsequent *implication inference*. High-level pseudocode for this subcomponent is provided in Algorithm 1.

Algorithm 1 Predicate Inference

```
1: procedure InferenceEngine($T_r$, ptrns, params)
2:     $R$ = set()
3:     $g^r_{i-1}$ = None
4:     for $g^r_i$ in $T_r$ do
5:         // --- Instantiate new relations ---
6:         $n$ = findNewEntities($g^r_i$, $g^r_{i-1}$)
7:         pairs = findPairings($n$, $g^r_i$)
8:         $R_k$ = instantiateRels($n$, pairs, ptrns)
9:         $R$.update($R_k$)
10:        // --- Evaluation and refinement ---
11:        for $r_i$ in $R$ do
12:            $\omega$ = evaluate($r_i$, $g^r_i$, params)
13:            $r_i$.evals[$g^r_i$] = $\omega$
14:            if $\omega$ == False then
15:                $r^<_i$ = refine($r_i$, $g^r_i$)
16:                $R$.add($r^<_i$)
17:                $r^<_i$.evals = refineEvals($r_i$.evals)
18:        $g^r_{i-1}$ = $g^r_i$
19:    $R_\top$ = extractTruePredicates($R$)
20:    return ($R$, $R_\top$)
```

For each reachability graph $g^r_i \in T_r$ (line 4), the engine starts by instantiating relation patterns with previously unseen permutations of entities from g^r_i (lines 6-9). To do so, this component retrieves new entities from g^r_i and constructs all new pairings of nodes. This is necessary to account for systems in which entity information is occasionally inaccessible (e.g. observations are collected by n local observers, information is only accessible within a subset of a map).

Example 3. In the traffic scenario, consider the first reachability graph in the trace, g^r_0. Since all permutations of single and paired entities have not been used to instantiate patterns, every pattern is filled with a corresponding set of entities. Consider the following pattern: $ENTITY.neighbors.size \geq CONST$. To instantiate relations from this pattern, the engine first replaces the $ENTITY$ token with an entity name (e.g. for car_1: $car_1.neighbors.size \geq CONST$). Afterward, the $CONST$ token is replaced by evaluating the new left-hand term, $car_1.neighbors.size$. If car_2 and car_3 are adjacent to car_1 in the neighbor graph g^r_0, then the component determines that $car_1.neighbors.size = 2$ at timestep 0. Upon replacing the $CONST$ token with this value, the fully-instantiated relation is obtained: $car_1.neighbors.size \geq 2$.

After all new instantiations have been made from the current reachability graph g^r_i, all relations are evaluated under g^r_i as $True, False$, or $Unknown$ (line 12). The result is $Unknown$ when the relation cannot be evaluated, due to an entity's absence from the current observation. To enable the engine to later infer implications between relations, each relation contains a dictionary that stores the evaluation from each timestep. This mapping is of the form $\Omega : (R \times G^r) \rightarrow True|False|Unknown$, where R is the set of all generated relations and G^r is the set of all reachability graphs. Finally, if a relation is evaluated as $False$, then a refined version is generated to make it $True$ under g^r_i. This relaxed relation is then marked as $True$ in its evaluations for all graphs g^r_k in which its parent was evaluated as $True$, in addition to the current graph g^r_i (lines 15-17).

Example 4. In the traffic scenario, assume that the graph associated with the subsequent timestep, g^r_1, shows that car_2 is no longer a neighbor of car_1. In this case, since car_3 is the only entity that is adjacent to car_1 in g^r_1, the component finds that $car_1.neighbors.size = 1$. This means that the relation that was instantiated at the previous step, $car_1.neighbors.size \geq 2$, is $False$ and should be relaxed. The right-hand term is updated by re-evaluating the left-hand term, thereby generating a newly refined relation: $car_1.neighbors.size \geq 1$.

After this iterative process of instantiation, evaluation, and refinement, the set of all relations are passed to the *Implication Inference* subcomponent as R, and the set of relations that were never violated are stored as $R_\top \subseteq R$.

3) Implication Inference: The goal of the *implication inference* subcomponent is to use the evaluations saved from each relation (at each timestep) to infer implications between pairs of relations in R. Prior to forming such implications, this subcomponent reduces the input space by excluding relations that were evaluated as $True$ or $False$ for all timesteps. In essence, this step removes predicates that would $weaken$ the implication and thereby introduce noise into the subcomponent's output.

For each eligible pair of predicate relations $(r_j,\ r_k)$, this subcomponent checks the implication between their evaluations $\Omega(r_j, g^r_i) \Rightarrow \Omega(r_k, g^r_i)$ for each timestep i. If a contradiction (i.e. $True \Rightarrow False$) is found at any timestep, the implication is flagged such that the subsequent *Relation Generalizer* will not report its general form. Afterward, all generated implications are passed to the *Relation Generalizer* subcomponent as the set R_{imp}.

4) Relation Generalizer: The objective of the *Relation Generalizer* is to make generalizations about various *groups* of entities for the extraction of broader spatial relations. Generalizations are made across all entities by default, but the user may provide specific groupings (e.g. by type, behavioral class, region), given that the associated entity classifications are provided in the trace (e.g. n_j.class = "emergency"; n_k.class = "bus") For each relation in $R_\top$ and R_{imp}, the general form is extracted by replacing each entity name with its group name. Next, these group names are replaced with every viable permutation of its members. The generalized

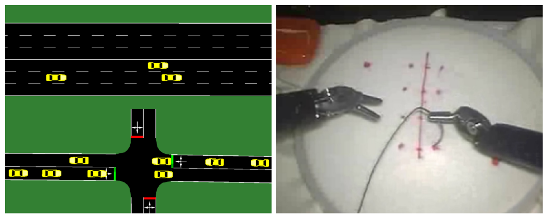

Fig. 3. *SUMO* highway (top left) and intersection (bottom left), JIGSAWS "Suturing" procedure (right).

relation is only reported if none of its instantiations are violated at any timestep in the sequence of reachability graphs. Finally, the set of generalized true predicates ($R_{\top}^{G}$) and the set of generalized implication relations (R_{imp}^{G}) are passed through the filter.

C. Filter

Prior to reporting, the single entity-level predicates and generalized relations are filtered by means of a *tautological model* and *logical subsumption* and *exclusion*. The tautological model is a lattice which describes which neighborship distinctions are contained within another through *ancestor/descendant relationships*. By default, such lattice informs the engine that $ENTITY.Neighbors \subseteq ENTITY.Neighborhood$. Alternatively, the user can provide more complex distinctions between sets of neighbors, which is beneficial to remove relations that would not be caught by standard logical subsumption.

After these tautological relations are removed, the filter checks whether each relation can be logically subsumed by another. For instance, in the case that an implication ($A \Rightarrow B$) and its contrapositive ($\neg B \Rightarrow \neg A$) are part of the filter's input, the latter is removed by logical equivalency. Additionally, greater specificity is favored by the filter through exclusion. For example, if relations $ENTITY.AllNeighbors.size = 3$ and $ENTITY.AllNeighbors.size \leq 3$ are passed through filter, then the latter is removed. After the filtering step, all remaining relations are reported in the set R_k.

IV. Study

Through this study, we aim to assess the power and expressiveness of SpRInG by providing evidence that it can uncover relevant spatial relations over two distinct and complex physical systems, with minimal input required from the user. We seek to answer two research questions:

- **RQ1**: How effective is SpRInG in its ability to capture *new* and *existing* spatial relation specifications from traces?
- **RQ2**: How capable is SpRInG in its ability to produce relations that reflect contextual changes?

We answer RQ1 through a study of simulated traffic scenarios and answer RQ2 via a study of surgical robot system. We select the inference parameters and relationships guiding spatial abstraction based on the system, while the core inference framework remains unchanged. Full implementation details for each study are provided in our GitHub repository.

A. RQ1: Traffic

To answer our first research question, we target the *SUMO* (*"Simulation of Urban MObility"*) traffic system [21], due to its frequent use by roboticists to analyze AV behaviors in traffic flow [22, 23].

1) Setup: To capture diverse traffic behaviors, we construct *highway* and *intersection* scenarios (see Figure 3). We used the *SUMO Traffic Control Interface (TraCI)* to extract traces of observations that contain the *current lane, pose, velocities, turn signals, brake, etc.* for each vehicle.

2) Instantiation of Approach: We, as the user, define a set of domain-specific notions of neighbor to better capture reachability between vehicles. More specifically, we extract and utilize lane data from *SUMO* to extract vehicles from the same lane (i.e. *leaderNeighbors* and *followerNeighbors*) and vehicles that block adjacent lanes (i.e. *leftNeighbors* and *rightNeighbors*). These distinctions form each vehicle's set of neighbor definitions, and were included in the tautological model for filtering.

3) Evaluation: We seek to show that SpRInG is expressive enough to generate relations that capture existing system specifications. In this study, SpRInG has no prior knowledge of the structure or contents of existing specifications. After compiling a small sample of real *SUMO* specifications, we examined the output of SpRInG applied to traces extracted from the two traffic scenarios (see Figure 3). A sample of real and inferred specifications are provided in Table I.

These results demonstrate that SpRInG can reasonably capture underlying relationships that govern *SUMO* vehicle behavior. Since each original English specification is now paired with a formula in first-order logic, these can be mathematically checked and monitored. We find that inferred relations R_{1A}, R_{1B}, and R_2 accurately reflect their associated *SUMO* specifications, and are unlikely to be violated by the system. Inferred relation R_3, however, is weaker in that a counterexample is likely to be found under more observations (e.g. it would be violated if the traffic light changes to *red* while a vehicle is yielding within the intersection). To capture a more robust version of this specification, the user would need to provide a custom pattern of greater specificity and complexity.

Next, we explore the power of SpRInG to uncover useful system specifications that were previously unstated. We determine that an inferred specification is *useful* (i.e. a likely "true positive") if it is likely to never be violated from similar scenarios. To evaluate the *usefulness* of the specifications reported by SpRInG, we collect two *clusters* of traces: C_1 includes traces from three highway scenarios (comprised of 3, 4, and 5 lanes, while all other simulation parameters are held constant) and C_2 includes traces from three intersection scenarios in which the only modified parameter is a seed to randomize vehicle departures (timestep, lane), and behaviors (maximum acceleration and deceleration, etc.) for the traffic generator. We then make inferences over each cluster's traces with SpRInG, and compute the rate of "likely true positives". This rate, $TP(C_i)$, is the proportion of inferred relations that were never violated by any of the traces in cluster C_i. We

SUMO Specifications	Learned Relations
[$Reng_1$]:*"A vehicle may only change its lane if there is enough physical space on the target lane..."*	[R_{1A}]: vehicle.LaneChangeLeft ⇒ vehicle.LeftNeighbors.size == 0 [R_{1B}]: vehicle.LaneChangeRight ⇒ vehicle.RightNeighbors.size == 0
[$Reng_2$]: *"[When an emergency vehicle has its lights on,] traffic participants [must let] the emergency vehicle pass."*	[R_2]: ambulance.sirenOn ⇒ ambulance.LeaderNeighbors.size == 0
[$Reng_3$]: *"[Vehicles] are not permitted to enter the intersection... if there is a red traffic light."*	[R_3]: vehicle.inJunction ⇒ vehicle.NextTLState == 'green'

TABLE I

SPECIFICATIONS PROVIDED BY SUMO DOCUMENTATION (LEFT) AND RELATIONS INFERRED BY SpRInG (RIGHT).

Relations reported by s_{novice}, violated by s_{expert}	New relations in s_{expert}, not reported by s_{novice}.
[R_A]: (*tissue* ∈ *robot_right*.Neighbors) ⇒ (*robot_right*.velocity ≤ 0.05)	[R_E]: (*robot_right*.open) ⇒ (*robot_right* ∉ *tissue*.Neighbors)
[R_B]: (*tissue* ∈ *robot_right*.Neighbors) ⇒ (*robot_right*.rot_velocity ≤ 3.10)	[R_F]: *"Pushing needle through tissue"* ⇒ (*tissue*.Neighbors.size == 2)
[R_C]: *"Orienting needle"* ⇒ (*robot_left* ∈ *robot_right*.aboveNeighbors)	[R_G]: *"Orienting needle"* ⇒ (*tissue* ∈ *robot_left*.belowNeighbors)
[R_D]: (*robot_right*.open) ⇒ (*robot_left* ∈ *robot_right*.belowNeighbors)	[R_H]: (*robot_left*.open) ⇒ (*robot_left*.Neighbors.size == 2)

TABLE II

DIFFERENCE IN REPORTED RELATIONS BETWEEN *novice* AND *expert* OPERATORS.

observe that cluster C_1 reports a total of 383 generalized relations with $TP(C_1) = 79.4\%$, and cluster C_2 reports 574 generalized relations with $TP(C_2) = 93.6\%$.

We now provide examples of previously unstated specifications that were reported by SpRInG as "likely true positives" over the scenario clusters. First, it is reported that $(\exists[\mathit{VEHICLE.LeaderNeighbors}, v] : v.\mathit{in\_junction}) \Rightarrow (\mathit{VEHICLE.NextTLState} == \text{'}\mathit{green}')$. This relation states that if an ego vehicle observes one of its lead vehicles v to be within a traffic junction, then it should also observe that the traffic light is green. This follows our intuition for two reasons: we would expect a vehicle to only enter a junction when it detects a green traffic light, and its followers should also observe that the light is green. It is also reported that $(\mathit{VEHICLE.in\_junction}) \Rightarrow (\forall[\mathit{VEHICLE.LeaderNeighbors}, v] : v.\mathit{brake} == \mathit{False})$, which states that if an ego vehicle is within a junction, then none of its leaders are braking. This also follows our intuition, since vehicles will rarely decelerate through intersections, particularly in cases of light traffic.

These results demonstrate that SpRInG is powerful enough to uncover known and unknown specifications in the context of the SUMO traffic system.

B. Case Study 2: da Vinci Surgical System

To answer our second research question and to demonstrate the broad applicability of our approach, we evaluate SpRInG on its ability to learn spatial relations from the *JHU-ISI Gesture and Skill Assessment Working Set* (*JIGSAWS*) dataset. *JIGSAWS* contains data collected at 30Hz from a *da Vinci Surgical System* (*dVSS*), teleoperated by eight surgeons with varying levels of experience. Each surgeon performed five separate trials of three common procedures (*needle-passing*, *suturing*, and *knot-tying*). A surgeon with extensive relevant experience evaluated the technical skill exhibited in each trial, by means of a Global Rating Score (GRS). Finally, each timestep was manually labeled with the current gesture (e.g. *"Orienting needle"*) [24].

1) Setup: To capture a broad range of gestures, we selected the *suturing* procedure as our target for this study. We first convert the raw data into the JSON format that is accepted by the *Spatial Encoder*. As a pre-processing step, to form a much richer trace of reachability graphs, we modified the dataset to include the approximate position of the tissue. Our final traces contain information regarding the *position, rotational and linear velocities,* and *gripper angle* for each of the robot manipulators, the *location* of the tissue, and the current *gesture*.

2) Instantiation of Approach: To infer relations that more precisely capture the vertical configuration of the manipulators, we utilize entity positions on the *z-axis* to distinigush between *aboveNeighbors* and *belowNeighbors*. We then include these distinctions in the set of neighbor definitions for each robot manipulator and in the tautological model.

3) RQ2: Contextual Changes: We evaluate SpRInG in its ability to capture changes in relations between two similar scenarios. To do so, we compare relations learned from the trial with the lowest and highest GRS scores (s_{novice} and s_{expert}, respectively), over the same suturing procedure. We provide a comparison of the relations reported by SpRInG over these scenarios in Table II. The left column provides four spatial relations inferred from s_{novice}, but violated by s_{expert}, and the right column contains relations that were inferred from s_{expert}, but violated by s_{novice}. These results reveal that the expert operator exhibits quicker movements than the novice while the suture is near the tissue (R_A, R_B) and that there are differences between operators in the vertical configuration of the robot manipulators during the procedure (R_C, R_D). We also find that, unlike the novice operator, the expert passes the suture to the right hand away from the tissue (R_E) and uses both arms to aid with insertion (R_F), among other distinctions in their technique (R_G, R_H).

V. CONCLUSION

We present SpRInG, a framework for the unsupervised inference of system specifications that characterize the relationship between the sensed distribution of objects and resulting robot behaviors. Our studies show that SpRInG can learn existing specifications and uncover previously unstated specifications in complex systems, both real and simulated.

References

[1] E. Asarin et al. "Parametric Identification of Temporal Properties". In: *International Conference on Runtime Verification* 7186 (Sept. 2011), pp. 147–160.

[2] S. Mohammadinejad, J. Deshmukh, and L. Nenzi. "Mining Interpretable Spatio-Temporal Logic Properties for Spatially Distributed Systems". In: *Automated Technology for Verification and Analysis* 12971 (Oct. 2021).

[3] M. Ernst et al. "Dynamically Discovering Likely Program Dynamically Discovering Likely Program Invariants to Support Program Evolution". In: *IEEE Transactions on Software Engineering* 27.2 (Feb. 2001).

[4] T. Nguyen et al. "DIG: A Dynamic Invariant Generator for Polynomial DIG: A Dynamic Invariant Generator for Polynomial and Array Invariants". In: *ACM Transactions on Software Engineering and Methodology* 23.4 (Aug. 2014).

[5] Mark Gabel and Zhendong Su. "Javert: Fully Automatic Mining of General Temporal Properties from Dynamic Traces". In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. SIGSOFT '08/FSE-16. Atlanta, Georgia: Association for Computing Machinery, 2008, pp. 339–349. ISBN: 9781595939951. DOI: `10.1145/1453101.1453150`. URL: `https://doi.org/10.1145/1453101.1453150`.

[6] S. Silvetti et al. "A Robust Genetic Algorithm for Learning Temporal Specifications from Data". In: *QEST*. 2018.

[7] F. Fages and A. Rizk. "From Model-Checking to Temporal Logic Constraint Solving". In: *Conference on Principles and Practice of Constraint Programming* (Jan. 2009), pp. 319–334.

[8] G. Chen, Z. Sabato, and Z. Kong. "Active Requirement Mining of Bounded-Time Temporal Properties of Cyber-Physical Systems". In: *IEEE 55th Conference on Decision and Control* (2016), pp. 4586–4593.

[9] Marat Boshernitsan, Roongko Doong, and Alberto Savoia. "From Daikon to Agitator: Lessons and Challenges in Building a Commercial Tool for Developer Testing". In: *Proceedings of the 2006 International Symposium on Software Testing and Analysis*. ISSTA '06. Portland, Maine, USA: Association for Computing Machinery, 2006, pp. 169–180. ISBN: 1595932631.

[10] David Schuler, Valentin Dallmeier, and Andreas Zeller. "Efficient Mutation Testing by Checking Invariant Violations". In: *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis*. ISSTA '09. Chicago, IL, USA: Association for Computing Machinery, 2009, pp. 69–80.

[11] Toh Ne Win and Michael Ernst. "Verifying distributed algorithms via dynamic analysis and theorem proving". In: *Technical Report MIT-LCS-TR-841* (2002).

[12] Brian Demsky et al. "Inference and Enforcement of Data Structure Consistency Specifications". In: *Proceedings of the 2006 International Symposium on Software Testing and Analysis*. ISSTA '06. Portland, Maine, USA: Association for Computing Machinery, 2006, pp. 233–244.

[13] H. Jiang, S. Elbaum, and C. Detweiler. "Inferring and Monitoring Invariants in Robotic Systems". In: *Autonomous Robots* 41.4 (Apr. 2017), pp. 1027–1046.

[14] M. Ma et al. "Predictive Monitoring with Logic-Calibrated Uncertainty for Cyber-Physical Systems". In: *ACM Transactions on Embedded Computing Systems* 20.5 (Oct. 2021), pp. 1–25.

[15] Meiyi Ma et al. "STLnet: Signal Temporal Logic Enforced Multivariate Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 14604–14614. URL: `https://proceedings.neurips.cc/paper/2020/file/a7da6ba0505a41b98bd85907244c4c30-Paper.pdf`.

[16] Xin Qin and Jyotirmoy V. Deshmukh. "Clairvoyant Monitoring for Signal Temporal Logic". In: *Formal Modeling and Analysis of Timed Systems*. Ed. by Nathalie Bertrand and Nils Jansen. Cham: Springer International Publishing, 2020, pp. 178–195. ISBN: 978-3-030-57628-8.

[17] L. Nenzi et al. "Qualitative and Quantitative Monitoring of Spatio-Temporal Properties with SSTL". In: *Logical Methods in Computer Science* 14 (Oct. 2018), pp. 1–38.

[18] E. Bartocci et al. "Monitoring Mobile and Spatially Distributed Cyber-Physical Systems". In: *Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*. Vienna, Austria: Association for Computing Machinery, 2017, pp. 146–155.

[19] I. Haghighi et al. "SpaTeL: A Novel Spatial-Temporal Logic and Its Applications to Networked Systems". In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. HSCC '15. Seattle, Washington: Association for Computing Machinery, 2015, pp. 189–198.

[20] M. Ma et al. "SaSTL: Spatial Aggregation Signal Temporal Logic for Runtime Monitoring in Smart Cities". In: *ACM/IEEE International Conference on Cyber-Physical Systems* 2 (Apr. 2020), pp. 51–62.

[21] Pablo Alvarez Lopez et al. "Microscopic Traffic Simulation using SUMO". In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL: `https://elib.dlr.de/124092/`.

[22] Pedro Fernandes and Urbano Nunes. "Platooning of autonomous vehicles with intervehicle communications in SUMO traffic simulator". In: Oct. 2010,

pp. 1313–1318. DOI: 10 . 1109 / ITSC . 2010 . 5625277.

[23] Qiong Lu et al. “The impact of autonomous vehicles on urban traffic network capacity: an experimental analysis by microscopic traffic simulation”. In: *Transportation Letters* 12.8 (2020), pp. 540–549. DOI: 10 . 1080 / 19427867 . 2019 . 1662561. eprint: https : / / doi . org / 10 . 1080 / 19427867 . 2019.1662561. URL: https://doi.org/10.1080/19427867.2019.1662561.

[24] Yixin Gao et al. “JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) : A Surgical Activity Dataset for Human Motion Modeling”. In: Modeling and Monitoring of Computer Assisted Interventions (M2CAI) – MICCAI Workshop, 2014.