SpRInG: A Spatial Relation Inference Generator for Robot Systems

---

A

Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Master of Science

by

Christopher Morse

May 2023

APPROVAL SHEET

This

Thesis

is submitted in partial fulfillment of the requirements
for the degree of

Master of Science

Author: Christopher Morse

This Thesis has been read and approved by the examining committee:

Advisor: Sebastian Elbaum

Advisor:

Committee Member: Matthew Dwyer

Committee Member: Lu Feng

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

May 2023

SpRInG: A Spatial Relation Inference Generator for Robot Systems

Christopher Morse

May 2023

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 specify the underlying relations between sensed object spatial distributions and robot behaviors. This thesis aims 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 from traces 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 provide the structure to infer, from system traces, the connection between such relationships and robot behaviors. We show that SpRInG can automatically infer spatial relations over two distinct domains: autonomous vehicles in traffic and a surgical robot. Our results demonstrate the power and expressiveness of SpRInG, in its ability to learn existing specifications as machine-checkable first-order logic, uncover previously unstated specifications that are rich and insightful, and reveal contextual differences between executions.

Acknowledgments

I extend my sincerest gratitude to my advisor, Dr. Sebastian Elbaum, for his unwavering support and insight on this project. I am also extraordinarily thankful for the help from my co-authors and defense committee members, Dr. Lu Feng and Dr. Matthew Dwyer. I would also like to thank Dr. Carrick Detweiler from the University of Nebraska - Lincoln (NIMBUS Laboratory) and Dr. Junaed Sattar from the University of Minnesota (IRV Laboratory) for giving me opportunities to explore scientific research. Finally, I am endlessly grateful for the support from my colleagues in the LESS Laboratory at the University of Virginia, my friends, and my family.

Contents

Chapter 1

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 1-1). In this scenario, we expect an AV to change lanes only if it senses that there are no vehicles blocking its target lane. We may also observe that an AV will decelerate to allow a nearby vehicle to pass, or operate with more caution in dense traffic (e.g. decelerates, performs fewer lane changes). Furthermore, we may observe that a sudden deceleration by a lead vehicle would cause the AV to brake.

As robot systems and their operating environments grow in complexity, the underlying relation between the spatial distributions 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, so manually defining relevant relationships between all entities and variables

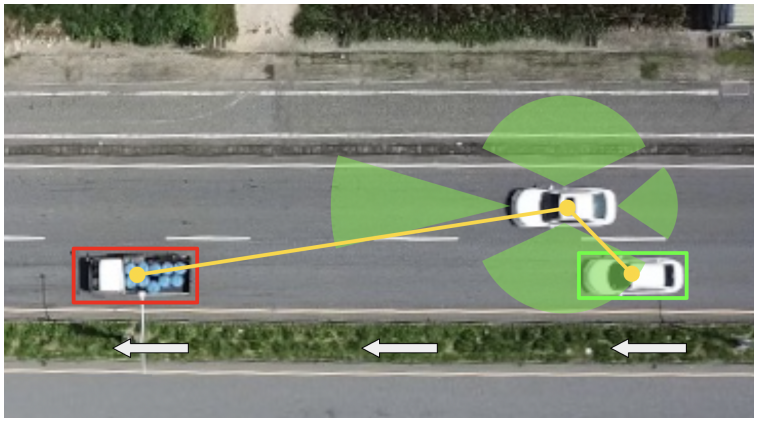

Figure 1-1: Autonomous vehicles operating in a traffic scenario. The boxed vehicles (bottom lane) are those that are detected by the AV in the top lane, and the yellow lines represent their sensed distances. If a vehicle is detected within one of the shaded green regions around the AV, then it is determined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

quickly becomes infeasible. Second, the increasing number of learned components in robots tends to obscure such spatial relationships. In the traffic scenario, the behavior of each AV may not be explicitly stated in the code, but rather governed 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, complex scenarios introduce highly unpredictable factors, including the variability of the operating environment (e.g. weather, road conditions), 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 behaviors of robot systems, leading many to remain unspecified. Yet, since the misbehavior of robots can lead to catastrophic outcomes, it is important to define these specifications

to enable the verification and validation of such systems. To address this challenge, we aim to automatically infer relations that help to characterize how sensed object spatial distributions and robot behaviors are related, by leveraging the increasingly rich sensor data collected by deployed robots.

In the context of the traffic scenario, the following is an example of a relation that we would want to capture:

- *If an AV senses a nearby vehicle in its left lane, then the AV would* ***not perform a left lane change****.*

This relation defines expected lane change behavior, and would likely be included as a safety specification for an AV. When operating safely, we would expect this relation to never be violated. In Figure 1-2, we provide two sequences of traffic scenarios. The first outlines the expected lane change behavior, which abides by the relation outlined above, and the second demonstrates a violation.

We may also want to capture relations like:

- *If an AV senses that its lead vehicle decelerates, then the AV will* ***decelerate****.*
- *When an AV senses a red traffic light ahead, then the AV will* ***stop****.*
- *If an AV senses an active emergency vehicle, then it will* ***pull over to the side of the road****.*

Such inferred relations have two desirable attributes. First, they possess a higher level of abstraction than the robot code implementation, one that explicitly connects object spatial distributions and the physical behaviors of the robots. Second, they can be inferred with no prior knowledge of system specifications, as they are learned solely from system traces of sensing and actuation data.

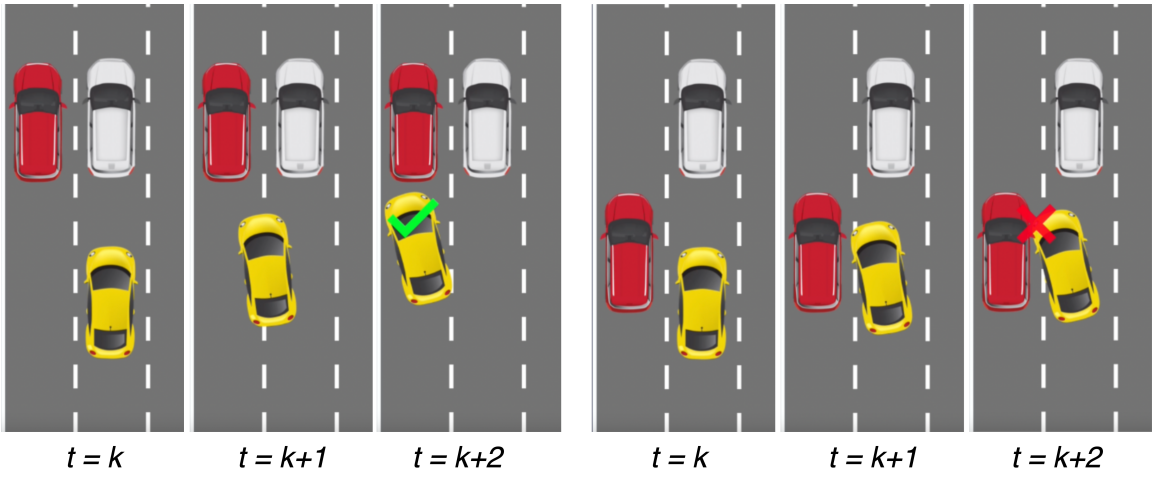

Figure 1-2: Example of expected AV lane change behavior (left) and unsafe AV lane change behavior (right).

1.1 Thesis Objective

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, and then utilizes 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.

Another objective of this thesis is to provide an extensive and detailed overview of SpRInG. While portions of this thesis were formed into a paper that will be appearing

at the 2023 IEEE International Conference on Robotics and Automation [3], many details were excluded to satisfy the page limit constraint. For this reason, this thesis is an extension of the conference paper, as it provides examples and details of our approach, additional figures, implementation details, and a discussion of avenues for future work. The primary contributions of this work are:

- A novel framework (SpRInG) for characterizing the spatial relationships between sensed object distributions and physical robot behaviors.
- A publicly available implementation of SpRInG, along with additional documentation and implementation details.[1]
- An assessment of SpRInG over two distinct domains, to demonstrate its ability to automatically generate insightful spatial relations over diverse systems.

1.2 Thesis Structure

The remainder of this thesis will be structured as follows: In Chapter 2, we discuss related work in the field of relation inference and usage. Chapter 3 provides a thorough description of the inner workings of the SpRInG framework, along with a running example from the traffic scenario. We then explore two research questions through a study of two distinct systems in Chapter 4, to demonstrate the power and expressiveness of SpRInG. In the final chapter, we provide ideas for future extensions and applications of the SpRInG framework.

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

Chapter 2

Related Work

The inference of spatial relations for robot systems spans across several fields of study. These include the inference from traces, the characterization of spatial relationships, and the way in which spatial models are encoded.

2.1 Inference from Variable-Value Traces

The work on automated inference 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 [4], 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 [5], 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 [6], or genetic algorithms to infer signal temporal logic (STL) parameters [7].

Second, by the information that the user must provide to the inference process. In general, uncovering more complex relations (e.g. multi-variate, multi-entity, temporal) requires a more expensive approach, since the analysis must explore a much larger set of potential relationships. This implies that, in order to be tractable, approaches that can encode these sophisticated relationships tend to require guiding information from the user. This information provides the inference process with the contents or structure that are likely to form desirable relations. 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 [8, 9, 1].

Third, by the way in which inferred relations are used. Software developers have used such relations for testing [10, 11], verification [12], debugging [13], robot system monitoring [14], predictive monitoring [15, 16, 17], and to characterize uncertainty in robot systems [15].

2.2 Characterization of Spatial Relationships

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 [18], STREL [19], SpaTeL [20], and SaSTL [21]), 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 capture only a small subset of complex specifications, relative to modern spatial logics [21]. Even further, our approach can extract spatial relationships that are much richer than distance bounds on spatial operators. Second, their approach requires the user to have substantial domain knowledge to provide patterns and parameter ordering. In contrast, our approach addresses these issues and offers a greater level of automation and control. By automatically instantiating a generated library of patterns while permitting the user to provide their own, our approach balances automated discovery with pattern-query specificity.

2.3 Spatial Model Encodings

Another core limitation for existing methods of characterizing spatial relationships, in the context of mobile robotics, is their modeling of spatial relationships. In the approach discussed above [2], the authors use *minimum-spanning graphs* as their spatial model, which preserves connectivity between nodes. This poses an assumption, however, that does not hold for mobile robots. For example, if the spatial relationship between two AVs is solely determined through their sensing equipment (e.g. cameras,

LiDAR), failure to detect one another (e.g. out of sensor range, an obstruction, etc.) should disconnect the nodes in the spatial model.

Alternatively, the *connectivity graphs* underlying Mobile Ad-Hoc Networks (MANETs) [22] do not assume spatial continuity, but still lack direct applicability to mobile robot systems for two reasons. First, MANETs are not multi-edge graphs, and therefore cannot account for complex relationships between nodes. For example, to capture the rich spatial relationships between two AVs, multiple edges are required to adequately describe their spatial relationship (e.g. sensed distance, relative orientation, relative velocity). Second, MANETs are bi-directional. However, due to the heterogeneity of sensing equipment and capabilities, the spatial models for mobile robot systems should also account for uni-directional relationships. For example, there may be a case in which one AV senses another, but not vice versa (e.g. due to sensor range, sensor placement, environmental factors).

Chapter 3

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 behaviors. The framework consists of three main components (see Figure 3-1). 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.

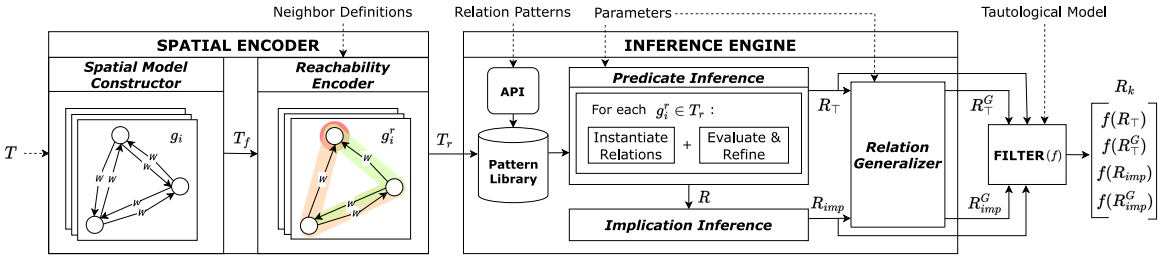

Figure 3-1: The SpRInG framework. Its three main components are shown: *Spatial Encoder*, *Inference Engine*, and the *Filter*. In the *Reachability Encoder* subcomponent, neighbor relationships between entities are visualized with highlighted edges.

3.1 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 [18, 19, 21] (see Figure 3-1: "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 vehicle, person). This information is stored as a set of variable-value pairs.

Example 1. Consider the traffic scenario with three AVs: car_1, car_2, and car_3. Each observation will contain state and perception information for each entity. Assume that the following state variables and perception information have been logged at each observation from car_1: *{(x,y) position, velocity, LiDAR detections}*, and $car_{2,3}$: *{(x,y) position, velocity, battery_level, camera detections}*.

The conversion of trace T into a sequence of graphs T_r is accomplished through two subcomponents: the *Spatial Model Constructor* and the *Reachability Encoder*.

3.1.1 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 $o_i \in T$ 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 (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 of attributes, $w_{jk} \in W$, that relates the paired entities by their relative state information and relevant perception information from n_j. For example, an edge's attribute set could contain the Euclidean distance between entities, relative orientations, sensing information (e.g. detections from LiDAR, cameras, IR), or shared occupancy regions in an environment (e.g. being in the same lane on a highway).

Example 2. Assume that this component is constructing a directed edge from car_1 to car_2 at observation o_i. To fill this edge's attribute set, the component tries to spatially relate this pair of entities using the data provided in the trace. First, relative state variables are computed by first finding the intersection of their state variable sets. Since *{(x,y) position, velocity, LiDAR detections}* $\cap$ *{(x,y) position, velocity, battery_level, camera detections}* = *{(x,y) position, velocity}*, this component computes and stores the Euclidean distance and relative velocity between the AVs in their corresponding edge. Next, since perception information from car_1 is provided within the trace, this component adds another edge attribute that indicates whether car_1 sensed car_2, along with other relevant information (e.g. detection confidence).

After iterating this process for all observations in trace T, each $o_i \in T$ has been abstracted into a fully-connected graph, $g_i \in T_f$.

3.1.2 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 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), it is reasonable to assign each entity a unique neighbor definition. To evaluate such a formula, this component considers the following mapping: $z : (\Phi \times 2^W) \rightarrow \mathit{True}|\mathit{False}$, 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 evaluated as either *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 3. In the traffic scenario, assume that the user provides the following neighbor definition 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.

After each subformula is evaluated, this subcomponent evaluates the entire neighbor definition. If the formula evaluates as *True*, then the associated edge is preserved in the graph. Otherwise, it is removed. 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 $e'_{jk} \in E'$ 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. Both definitions are optionally predicated under the *type* of neighbor (e.g. *leftNeighbors, rightNeighbors*).

The notion of reachability, which is entirely encoded within each reachability graph $g_i^r \in T_r$ through the *neighbor* and *neighborhood* relationships, are fundamental to the inference procedure.

3.2 Inference Engine

The goal of the inference engine component (see Figure 3-1: "Inference Engine") is to learn spatial relations from the sequence of reachability graphs. Since these graphs encode the *reachability* between entities through *neighborship*, the learned relations are based on the *neighbor* and *neighborhood* relationships.

3.2.1 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. In Table 3.1, we provide the full grammar supported by SpRInG.

This grammar consists of a start symbol (P_{main}), non-terminal symbols ($P_{1..6}$), terminal symbols (see "Supporting Definitions"), and production rules (see "Productions"). The final relation patterns will contain tokens *CONST* and *N*, that are later filled by the engine with a value or entity, respectively, in accordance with the provided grammar rules. The pattern structure and all other information are either provided by the user or constructed by the generator.

Productions
$\mathrm{P}_{\mathrm{main}} := \mathrm{P}_1 \mid \mathrm{P}_2 \mid \mathrm{P}_3 \mid \mathrm{P}_4 \mid \mathrm{P}_5 \mid \mathrm{P}_6$
$\mathrm{P}_1 := \mathrm{P}_{\mathrm{main}}$ OP $\mathrm{P}_{\mathrm{main}}$
$\mathrm{P}_2 :=$ CONST
$\mathrm{P}_3 := N$.VAR
$\mathrm{P}_4 := N$.Relation.Attribute
$\mathrm{P}_5 :=$ Aggregator($\mathrm{P}_{\mathrm{main}}$)
$\mathrm{P}_6 :=$ Quantifier(Members; $\mathrm{P}_{\mathrm{main}}$)
Supporting Definitions
OP: $\geq \mid \leq \mid == \mid \subset \mid \supset \mid \Rightarrow \mid \in \mid \notin$
CONST: $\mathbb{R} \mid$ *Class* $\mid$ *System.state*
N: *System.entity* $\mid$ *Group*
VAR: *System.variable*
Relation: *neighbors* $\mid$ *neighborhood* $\mid \varepsilon$
Attribute: *size* $\mid \varepsilon$
Aggregator: *min* $\mid$ *max* $\mid$ *avg*
Quantifier: $\forall \mid \exists$
Members: *Group* $\mid$ *Relation*

Table 3.1: Grammar for instantiating relation patterns.

Example 4. The relation pattern ($N.Neighbors.size \geq CONST$) is constructed by $(P_{main}) \rightarrow (P_1) \rightarrow (P_{main}\ \ OP\ \ P_{main}) \rightarrow (P_4\ \ OP\ \ P_2) \rightarrow (N.Relation.Attribute \geq CONST) \rightarrow (N.Neighbors.Size \geq CONST)$. Each $(a) \rightarrow (b)$ signifies that string (b) is a direct derivation from string (a), per the production rules.

3.2.2 Predicate Inference

The objective of the *Predicate Inference* subcomponent is to use the relation patterns, the sequence 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* subcomponent. High-level pseudocode for the *Predicate Inference* procedure is provided in Algorithm 1.

Algorithm 1 Predicate Inference

```
1:  procedure InferenceEngine(T_r, patterns, 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, patterns)
9:          R.update(R_k)
10:         // -- Evaluation and refinement --
11:         for r_i in R do
12:             ω = evaluate(r_i, g^r_i, params)
13:             r_i.evals[g^r_i] = ω
14:             if ω == 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_⊤ = extractTruePredicates(R)
20:     return (R,R_⊤)
```

For each reachability graph $g_i^r \in T_r$ (line 4), the engine starts by instantiating relation patterns with previously unseen permutations of entities from g_i^r (lines 6-9). To do so, this component retrieves new entities from g_i^r and constructs all new pairings of nodes. This is necessary to account for systems in which entity informa-

tion is occasionally inaccessible (e.g. observations are collected by n local observers, information is only accessible within a subset of a map).

Example 5. In the traffic scenario, consider the first reachability graph in the sequence, g_0^r. 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: $car_1.Neighbors.size \geq CONST$). Upon evaluation of the new left-hand term ($car_1.Neighbors.size$), the $CONST$ token is replaced with the resulting value. For example, if car_2 and car_3 are adjacent to car_1 in the reachability graph g_0^r, 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_i^r, all relations are evaluated under g_i^r 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. This may occur when the observations are not global (e.g. information is only accessible over a subset of the map, or for a subset of the system's entities). A pair of inference parameters are used to determine how these $Unknown$ evaluations should be handled, and would ideally be based on the user's assumption regarding the variability of the system and environment. The first parameter dictates the proportion of allowed $Unknown$ evaluations before the relation is removed. The second parameter restricts the number of consecutive $Unknown$ evaluations allowed.

To enable the engine to later infer implications between relations, each relation object contains a dictionary that stores the evaluation from each timestep. This mapping is of the form $\Omega : (R \times G^r) \rightarrow \mathit{True}|\mathit{False}|\mathit{Unknown}$, 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_i^r. This relaxed relation is then marked as *True* in its evaluations for all graphs g_k^r in which its parent was evaluated as *True*, in addition to the current graph g_i^r (lines 15-17).

Example 6. In the traffic scenario, assume that the graph associated with the subsequent timestep, g_1^r, shows that car_2 is no longer a neighbor of car_1. In this case, since car_3 is now the only entity that is adjacent to car_1, the component finds that $car_1.Neighbors.size = 1$. Therefore, the relation that was instantiated at the previous step, $car_1.Neighbors.size \geq 2$, is now *False* and should be relaxed. To do so, the left-hand term is re-evaluated and the right-hand term is updated, 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 $R_\top \subseteq R$.

3.2.3 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 $\Omega(r_j, g_i^r) \Rightarrow \Omega(r_k, g_i^r)$ 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}.

3.2.4 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. $n_j.\mathsf{class} = \text{“}\mathsf{ambulance}\text{”}$, $n_k.\mathsf{class} = \text{“}\mathsf{bus}\text{”}$). 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 relation is only reported if none of its instantiations are violated at any timestep in the sequence of reachability graphs.

Example 7. For example, consider the relation $(car_1.neighbors.size \geq 2) \in R_\top$. When generalizing over a *class* grouping, this relation would be converted to the generalized form $(CAR.neighbors.size \geq 2)$. The generalized form is reported only if it

is never evaluated as *False* for any permutation of the group's members. For example, for $(CAR.neighbors.size \geq 2)$ to be reported, the relation $(car_*.neighbors.size \geq 2)$ must never be false for any member of the *car* class.

Finally, the set of generalized true predicates ($R^G_\top$) and the set of generalized implication relations (R^G_{imp}) are passed through the filter.

3.3 Filter

Prior to reporting, all candidate relations are filtered by means of a *tautological model* and through *logical subsumption* and *exclusion*. The tautological model is a lattice that describes which neighborship distinctions are contained within another through *ancestor/descendant relationships*. By default, to help remove redundant relations, this lattice informs the filter that $ENTITY.Neighbors \subseteq ENTITY.Neighborhood$. Additionally, the user may provide more complex distinctions between sets of neighbors, which is beneficial to remove even more relations that would not be caught by standard logical subsumption (e.g. $ENTITY.LeftNeighbors \subseteq ENTITY.Neighbors$).

Example 8. In the traffic scenario, consider the case in which the user makes the distinction between *Front*, *Back*, *Left*, and *Right* neighbors, by defining the tautological model shown in Figure 3-2. With these relationships defined, the relation $(CAR.LeftNeighbors.size) \leq (CAR.Neighbors.size)$ is tautological, since the set in the left hand term (a *descendant*) is a subset of the set in the right hand term (an *ancestor*).

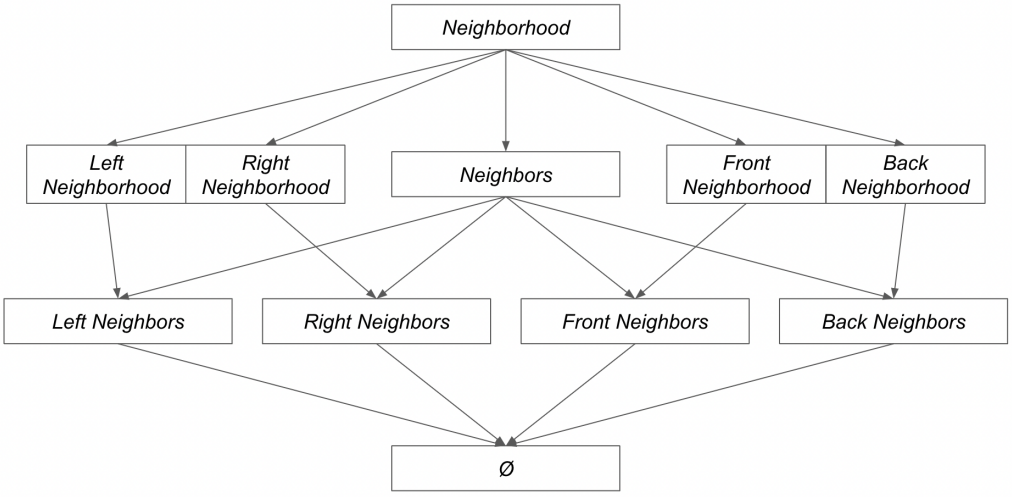

Figure 3-2: Example tautological model for the traffic scenario.

For implications that follow the pattern $(N.Neighbors.size\ OP\ CONST) \Rightarrow (N.Neighbors.size\ OP\ CONST)$, the filtering step examines restrictions on the predicates that were set by the tautological model. This process is guided by three rules to remove tautological implications. First, if an ancestor is in the antecedent and a descendant is in the consequent (i.e. $Ancestor \Rightarrow Descendant$), both predicates use the same operator "$\leq$" as OP, and their $CONST$ values are the same, then the implication is removed. Second, if the implication is of the form $Ancestor \Rightarrow Descendant$ and the antecedent has a $CONST$ value of 0, then it is removed. Third, if the implication has the form $Descendant \Rightarrow Ancestor$, the predicates share the same operator OP ("==" or "$\geq$"), and their $CONST$ values are equal but not zero, then it is removed.

Example 9. In the traffic scenario, consider the relation $(CAR.Neighbors.size == 0) \Rightarrow (CAR.FrontNeighbors.size == 0)$. Since the descendant's value is restricted by the ancestor, the implication would be removed by the filter (per the second rule). Additionally, the relation $(CAR.FrontNeighbors.size \geq 1) \Rightarrow (CAR.Neighbors.size \geq 1)$ would be removed by the third rule, since the ancestor's value is constrained by the descendant.

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 $ENTITY.Neighbors.size = 3$ and $ENTITY.Neighbors.size \leq 3$ are passed through filter, then the latter is removed. After the filtering step, all remaining relations are reported in the set R_k.

3.4 Tool Implementation

SpRInG was implemented in Python 3.7.6. The reachability graph abstraction, manipulation, and analysis, all heavily rely on the NetworkX package [23]. The trace, neighbor definitions (*optional*), and tautological model (*optional*), are all inputted into SpRInG as separate *JSON* files. Users also have the option to modify inference parameters through the command line or a *JSON* file to better configure the inference engine for their target system. While our approach does not generate highly complex relation patterns by default (e.g. nested implications), the user can alterna-

tively provide their own set of relation patterns. Most notably, our implementation is structured as independent subcomponents, which are easily modifiable. For example, since the inference engine only operates on a sequence of reachability graphs, the user can modify the *Spatial Encoder* as needed, independently from the inference engine. The generated relations are outputted as text files.

Chapter 4

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

We hypothesize that if the collected traces are rich enough to capture spatial patterns between entities, then both new and existing spatial relations, along with their underlying context, will be captured by SpRInG. We answer RQ1 through a study of simulated traffic scenarios and answer RQ2 through a study of surgical robot trials. We select the inference parameters and relationships guiding spatial abstraction based on the system, while the core inference framework remains unchanged.

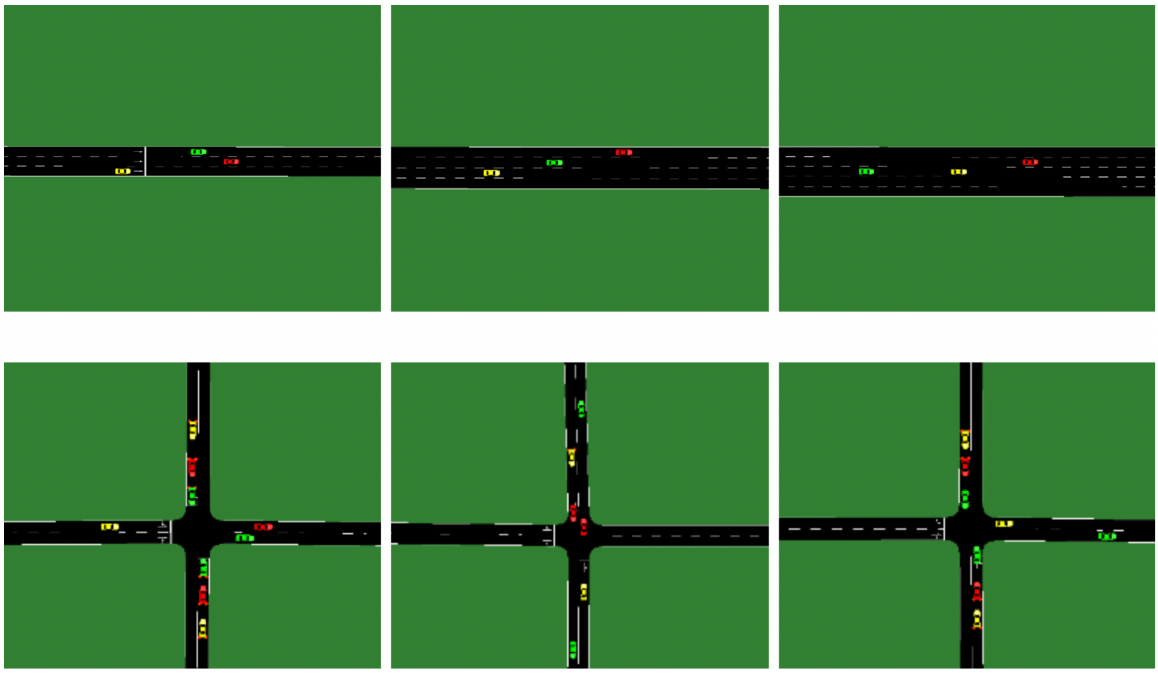

Figure 4-1: *SUMO* highway (top) and intersection (bottom) scenarios.

4.1 RQ1: Traffic

Understanding how AVs react to their sensed environment is essential for the safety and understanding of such systems. Due to their complexity, however, manually defining a robust set of specifications is often infeasible. Therefore, we seek to answer our first research question, which gauges the effectiveness of SpRInG to infer *new* and *existing* system specifications, from simulated traffic scenarios. We target the *SUMO* (*"Simulation of Urban MObility"*) system [24], which has seen frequent use in the analysis of AV behaviors in traffic [25, 26].

4.1.1 Setup

To capture diverse traffic behaviors, we construct *highway* and *intersection* scenarios (see Figure 4-1). 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. Each trace contains 1000 observations, collected at 1Hz.

4.1.2 Instantiation of Approach

We, as the user, define a set of domain-specific notions of *neighborship* to better capture the reachability between vehicles. For example, we may want to leverage the fact that a lead vehicle will likely have a different effect on an AV's behavior than a vehicle from an adjacent lane. More specifically, we extract and utilize lane data from *SUMO* to discern between vehicles from the same lane (i.e. *frontNeighbors* and *backNeighbors*) 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. In this case, the following neighbor definitions were defined for each vehicle. The leader and follower distances (20 meters, 2 meters) were determined by 5 and 0.5 times an approximate average vehicle length, in meters, respectively:

- $\Phi(\mathsf{vehicle}, front) = ((rellane == 0) \wedge (relroadID == 1) \wedge (-20 < distA < 0))$
- $\Phi(\mathsf{vehicle}, back) = ((rellane == 0) \wedge (relroadID == 1) \wedge (0 < distB < 2))$
- $\Phi(\mathsf{vehicle}, left) = ((rellane == -1) \wedge (relroadID == 1) \wedge ((distA > 0) \wedge (distB < 0)))$
- $\Phi(\mathsf{vehicle}, right) = ((rellane == 1) \wedge (relroadID == 1) \wedge ((distA > 0) \wedge (distB < 0)))$

The variables *rellane, relroadID, distA,* and *distB* are defined as follows:

- rellane = ENTITY1['laneID'] - ENTITY2['laneID']
- relroadID = 1 if ENTITY1['edge'] == ENTITY2['edge'] else 0
- distA = ENTITY1['frontBumperPos'] - ENTITY2['backBumperPos']
- distB = ENTITY1['backBumperPos'] - ENTITY2['frontBumperPos']

Since we can extract the ID (integer value) of the road and lane for all vehicles at each timestep, the *relroadID* and *rellane* variables extract information regarding whether the two vehicles are on the same road and their relative lane position. This information is used as part of the conditional to determine whether the vehicle should be considered as a *front, back, left,* or *right* neighbor. We can also extract the front and back bumper positions for each vehicle, which are effectively longitudinal distances along the vehicle's current road. With this information, the *distA* variable contains the longitudinal space between the ego vehicle's front bumper and the target vehicle's back bumper. This value (in meters) is negative when the target vehicle is ahead of it and and positive otherwise. In the neighbor definition for *front* neighbors, this value is used to ensure that a candidate lead vehicle is close enough to the ego vehicle to potentially influence its behavior. The *distB* variable is similar, but used to compute the distance to candidate *back* neighbors. Both of these variables form a condition in the neighbor definitions for *left* and *right* neighbors, as they are used to determine if there is any overlap in the longitudinal direction.

4.1.3 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 manually compiling a small sample of real *SUMO* specifications from its official documentation [24] , we examined the output of SpRInG applied to traces extracted from two traffic scenarios (see Figure 4-1). A sample of real and inferred specifications are provided in Table 4.1.

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.

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 by 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

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 siren on,] traffic participants [must let] the emergency vehicle pass."*	[R_2]: (ambulance.sirenOn) ⇒ (ambulance.FrontNeighbors.size == 0)
[$Reng_3$]: *"[Vehicles] are not permitted to enter the intersection... if there is a red traffic light."*	[R_3]: (vehicle.inJunction) ⇒ (vehicle.SensedTLState == "green")

Table 4.1: Specifications provided by SUMO documentation (left), associated relations inferred by SpRInG (right).

traces from three intersection scenarios in which the only modified parameter is a seed to randomize vehicle departures (e.g. timestep, lane) and behaviors (e.g. maximum acceleration and deceleration) 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:

$$TP(C_i) = \frac{\#\ \textit{Unique relations never violated by } C_i}{\#\ \textit{Unique relations reported by } C_i}$$

We 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 $(\exists[vehicle.FrontNeighbors, v] : v.in\_junction) \Rightarrow (vehicle.SensedTLState == \text{`}green')$. This relation states that if an ego vehicle observes one of its lead vehicles v to be within a traffic junction, then the ego vehicle 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 the traffic light is *green*, and its followers should also observe that the light is *green*. It is also reported that $(vehicle.in\_junction) \Rightarrow (\forall[vehicle.FrontNeighbors, v] : v.brake == False)$, 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 illustrate how SpRInG can be powerful enough to uncover known and unknown specifications.

4.2 RQ2: da Vinci Surgical System

To answer our second research question and to demonstrate the broad applicability of SpRInG, we evaluate 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*). Each

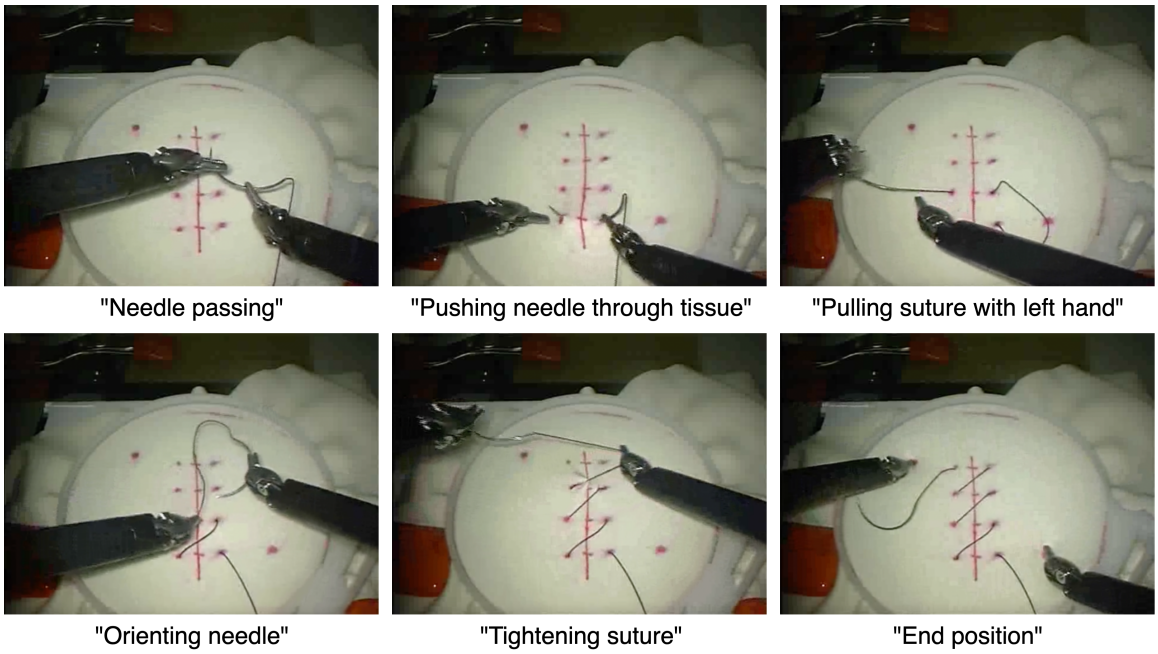

Figure 4-2: Examples of gestures from the JIGSAWS "Suturing" procedure.

trial was assigned a Global Rating Score (GRS) through a manual assessment of demonstrated skills. Finally, each timestep was manually labeled with the current gesture (e.g. *"Orienting needle"*) [27].

4.2.1 Setup

To capture a broad range of gestures, we selected the *suturing* procedure for this study (see Figure 4-2). We first convert the raw data into a format that is compatible with the *Spatial Encoder*. As a pre-processing step, to form a 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*.

4.2.2 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. The neighbor definitions for the robot manipulators and the tissue are provided below. Note that the tissue is always positioned below the manipulators along the z-axis, so distinctions for *above* and *below* in its neighbor definition were deemed unnecessary.

- $\Phi(\mathsf{manipulator}, above) = (dist < 0.055) \wedge (relz < 0)$
- $\Phi(\mathsf{manipulator}, below) = (dist < 0.055) \wedge (relz > 0)$
- $\Phi(\mathsf{tissue}) = (dist < 0.055)$

The variables *dist* and *relz* are defined as follows:

- $dist = d(ENTITY1, ENTITY2)$
- $relz = ENTITY1[`pos'_z] - ENTITY2[`pos'_z]$

The *dist* variable is the Euclidean distance (in meters) between two entities in 3D space, and the *relz* variable indicates the relative position along the z-axis. The values in the conditionals defined in the neighbor definitions were selected based on the

estimated length of the needle. These distinctions were included in the tautological model for filtering.

4.2.3 Evaluation

We evaluate SpRInG in its ability to reveal contextual differences between similar trials through its learned relations. To do so, we compare relations learned from trials 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 4.2. 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 needle 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

Relations reported by s_{novice}, violated by s_{expert}	Relations reported by s_{expert}, violated by s_{novice}
[R_A]: (*tissue* $\in$ *robot_right*.Neighbors) $\Rightarrow$ (*robot_right*.velocity ≤ 0.05)	[R_E]: (*robot_right*.open) $\Rightarrow$ (*robot_right* $\notin$ *tissue*.Neighbors)
[R_B]: (*tissue* $\in$ *robot_right*.Neighbors) $\Rightarrow$ (*robot_right*.rot_velocity ≤ 3.10)	[R_F]: *"Pushing needle through tissue"* $\Rightarrow$ (*tissue*.Neighbors.size == 2)
[R_C]: *"Orienting needle"* $\Rightarrow$ (*robot_left* $\in$ *robot_right*.aboveNeighbors)	[R_G]: *"Orienting needle"* $\Rightarrow$ (*tissue* $\in$ *robot_left*.belowNeighbors)
[R_D]: (*robot_right*.open) $\Rightarrow$ (*robot_left* $\in$ *robot_right*.belowNeighbors)	[R_H]: (*robot_left*.open) $\Rightarrow$ (*robot_left*.Neighbors.size == 2)

Table 4.2: Difference in reported relations between *novice* and *expert* operators.

that, unlike the novice operator, the expert passes the needle to the right while away from the tissue (R_E) and uses both arms to aid with insertion (R_F), among other distinctions in their techniques (R_G, R_H).

These results demonstrate that SpRInG can capture subtle contextual differences in its reported relations.

4.3 Summary of Findings

In this study, we examined our two research questions through simulated traffic data and real traces collected from a surgical robot. Our first research question targets the effectiveness of SpRInG at relation discovery. To show that SpRInG can uncover real specifications, we compared SpRInG-generated relations to specifications outlined in the official *SUMO* documentation. We then demonstrated that SpRInG can uncover valuable specifications that were not provided in this documentation. We showed that many of these relations are applicable to the system by checking them against a cluster of similar scenarios. Through our second research question, we demonstrated that the relations generated by SpRInG can reflect changes in context within a system. To do so, we used data from the *JIGSAWS* dataset to compare relations generated from novice and expert operators. While the task is the same between operators, SpRInG reveals subtle differences in the spatial configuration of the robot manipulators during the procedure. Overall, these results show that SpRInG is able to uncover relevant relations with ideal specificity over diverse systems, both real and simulated.

Chapter 5

Conclusion and Future Work

This thesis presents SpRInG, a framework for the unsupervised inference of system specifications that characterize the relationship between the sensed distribution of objects and resulting robot behaviors. The SpRInG framework accepts a trace of variable-value pairs from the user, converts the trace into a sequence of spatial models, instantiates a library of patterns, and infers relations over the sequence. The tool is highly configurable, with parameters to adjust the neighbor definitions, tautological model, and inference settings. We then demonstrate the power and expressiveness of SpRInG through two studies. The first demonstrates that SpRInG can learn existing specifications and uncover those that were previously unstated. The second shows that SpRInG can reveal subtle contextual differences between traces, through its reported relations.

We highlight that there are various paths for future extensions of SpRInG and its potential applications. These are provided below:

1. Most complex robot systems provide raw, unprocessed, and local data that are difficult to parse into a form that is meaningful for spatial relation inference.

To infer spatial relations from such systems, it is necessary to extend SpRInG to infer relations from more complex data types (e.g. images, point clouds). This would require a prefix to the SpRInG framework, to extract spatial models from such data.

2. The vast majority of mobile robot behaviors in response to sensed surroundings are not single-state, but often take a window of time to both react and respond. To account for this, it is necessary to extend SpRInG to infer spatial-temporal relations from event sequences. This would require SpRInG's *Pattern Library* to generate relation patterns that introduce linear temporal logic operators (e.g. *Next, Finally, Globally, Until, etc.*). This also requires infrastructural changes to the *Inference Engine* component, which could be modified to slide a window over a sequence of graphs, to check whether a given relation holds. Without restrictions on time bounds, there may be an intractably large search space. Therefore, to accelerate the inference process, it would likely be necessary for the user to provide some of this information to SpRInG.

3. Since real-world systems and their operating environments are highly uncertain (e.g. sensor noise, external conditions), the values in each observation have some level of noise. We would like to model this uncertainty and characterize robot behavior over such observations. This would require any uncertainty to be provided as input, and to be encoded in the reachability graphs constructed by the *Spatial Encoder* component. This also leads to three types of relationships between nodes: neighbors (certain), not neighbors (certain), and uncertain

neighbors. In the *Inference Engine*, this would require the edges to be evaluated with ternary logic, and the engine may output relations marked as *uncertain*. With this information, it may be worthwhile to study how an autonomous robot behaves when it is faced with high uncertainty in its detections.

4. For the safety of autonomous mobile robot systems, it is imperative to ensure that they can safely navigate through complex environments. Some existing motion planning frameworks utilize hierarchical “rulebooks" that govern how robots should navigate space [28]. However, these rules are manually defined, which means that they are a small subset of the full set of applicable rules for the system. For this reason, it would be valuable to use SpRInG to infer relations from existing data, and include them in such rulebooks.

Bibliography

[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] C. Morse et al. "A Framework for the Unsupervised Inference of Relations Between Sensed Object Spatial Distributions and Robot Behaviors". In: *2023 International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2023.

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

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

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Paresh Dhakan and Ronaldo Menezes. “The role of social structures in mobile AdHoc networks”. In: vol. 2. Mar. 2005, pp. 59–64. DOI: 10.1145/1167253.1167268.

[23] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[24] 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/.

[25] 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.

[26] 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.

[27] 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.

[28] Andrea Censi et al. "Liability, Ethics, and Culture-Aware Behavior Specification Using Rulebooks". In: *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE Press, 2019, pp. 8536–8542. DOI: 10.1109/ICRA.2019.8794364. URL: https://doi.org/10.1109/ICRA.2019.8794364.