Formalising Sensor Topologies for Target Counting*

Sven Linker
University of Liverpool
Department of Computer Science, Liverpool, UK
s.linker@liverpool.ac.uk

Michele Sevegnani
University of Glasgow
School of Computing Science, Glasgow, UK
michele.sevegnani@glasgow.ac.uk

We present a formal model developed to reason about topologies created by sensor ranges. This model is used to formalise the topological aspects of an existing counting algorithm to estimate the number of targets in the area covered by the sensors. To that end, we present a first-order logic tailored to specify relations between parts of the space with respect to sensor coverage. The logic serves as a specification language for Hoare-style proofs of correctness of the topological computations of the algorithm, which uncovers ambiguities in their results. Subsequently, we extend the formal model as a step towards improving the estimation of the algorithm. Finally, we sketch how the model can be extended to take mobile sensors and temporal aspects into account.

1 Introduction

The *target counting problem* is the computational task of estimating the total number of observable targets within a region using local counts performed by a set of networked sensors. With applications ranging from agriculture and wildlife protection to traffic control and indoor security, this problem has emerged as one of the most important applications of present and future sensor networks. Several different kinds of sensors may be deployed depending on the application domain. In this work, we focus on numeric photo-electric sensors capable of counting but not identifying targets in their vicinity, i.e., within their sensing range. Moreover, it is assumed that the exact position of the sensors and the geometry of their ranges is fully known. This information is referred to as the *topology* of a sensor network. The complexity of this problem lies in the fact that multiple sensors may be observing the same target if it is located within the intersection of their overlapping sensing ranges. This may lead to wrong estimates as duplicate observations, together with the inability to distinguish different targets, introduce over-counting. Various algorithms have been developed to mitigate this issue. For example, statistical information about target distribution is used in [15] and topological properties of the sensor network are exploited in [2]. The reader is referred to the recent work of Wu et al. [16] for a complete survey.

In our paper, we introduce a novel logical formalisation to reason about topologies arising from overlapping sensor ranges. We then use this logic to analyse how sensor topology can affect counting accuracy and to identify some hidden assumptions in SCAN [8], a geometry-based counting algorithm. We remark that the generality of our approach allows us to carry out the same kind of analysis for different target counting algorithms. Hence, we believe our work is an important first step towards the formal analysis of the topological properties crucial for the correct functioning of the systems we find in domains such as Smart Cities and IoT.

In this work, we draw from the experience of one of the authors in defining application-specific logics such as the spatial logic introduced in [11] for proving safety of traffic manoeuvres on a multi-lane highway. While the research on modal logics to reason about spatial properties has been on the

*This work is supported by the Engineering and Physical Sciences Research Council, under grant EP/N007565/1 (S4: Science of Sensor Systems Software).

D. Pianini and G. Salvaneschi (Eds.): First workshop
on Architectures, Languages and Paradigms for IoT
EPTCS 264, 2018, pp. 43–57, doi:10.4204/EPTCS.264.5

© S. Linker & M. Sevegnani
This work is licensed under the
Creative Commons Attribution License.

rise in the last decades [1], the focus was mostly on the general properties of the logics themselves, for example, decidability issues and complexity analysis of satisfaction. Furthermore, the basic entities of the semantics seem to be ill-fitting for our purposes. Consider the logic $S4$, e.g., which is often considered to be the prototypical logic capturing the notion of "nearness". Spatial models for $S4$ are build upon a topology, including an operator to compute the interior (or closure) of sets. The semantics of a formula is defined with respect to a point x in the topology, where the modal operator expresses that properties hold for all points "close" to x. However, in our case, we are not interested in properties of single points, or their immediate surroundings, but rather in the properties of whole regions of space, for example the intersections between sensor ranges. More recently, Ciancia et al. [4] used general closure spaces as semantics. However, the general issue with this approach for us remains: while overlapping regions can be modelled, it is cumbersome to reason about them. An approach more closely related to our intentions is the *Region Connection Calculus* (RCC) [13]. In RCC, the basic entities are convex connected regions of space, and the formulas describe the relations between them. Still, while it is easily expressible that two regions have a non-empty intersection, it is not possible to refer to only this intersection.

Our article is structured as follows. The next section gives a concise overview of the SCAN algorithm, with a particular focus on its topological aspects. Sect. 3 introduces STL, a logic to reason about the static topologies arising from overlapping sensor ranges in target counting scenarios. Topological temporal changes can be reasoned upon by using the temporal extension of STL defined in Sect. 4. Finally, Sect. 5 ends with conclusion and future work.

2 SCAN algorithm

SCAN is a geometry-based target counting algorithm introduced by Gandhi et al. [8] that computes an estimate of the number of targets by producing upper and lower bounds on the target population. These bounds are proved to provide worst-case guarantees for the target count for any choice of targets and sensor ranges. In this approach, each target is modelled as a point and sensor ranges are assumed to be convex shapes. This allows the algorithm to handle realistic sensing scenarios, such as those characterised by asymmetric, irregular, and anisotropic sensor ranges. Furthermore, it is assumed each sensor is capable of counting exactly the number of targets present in its range, i.e., ideal sensing,[1] the topology is fully known, and the spatial distribution of targets is unknown.

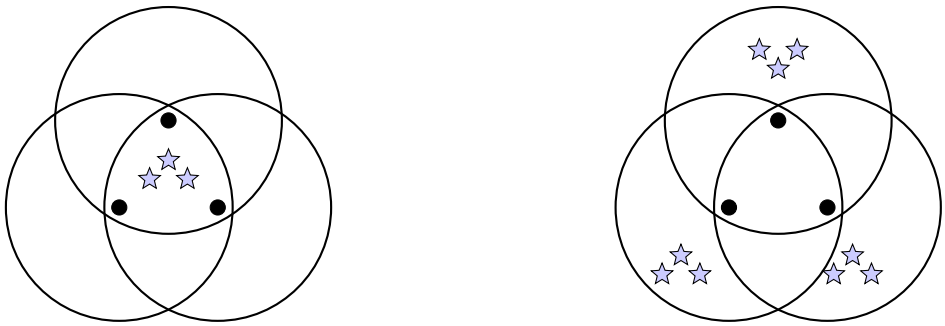
The key idea behind SCAN is that, in general, a constant-factor approximation of the true target count is impossible in two-dimensional space. However, an easy approximation within factor $\sqrt{m}$ is possible, where m is the maximum *degree of overlap* among sensor ranges. The degree of overlap is defined here as the maximum number of sensing ranges covering a point in the plane. A concise description of the various phases of SCAN is reported next.

2.1 Definition

Let S (sensors), C (counts), and R (ranges) be sets ranged over by s_i, c_i, r_i, respectively. We write c_i to indicate the number of targets detected by s_i. Similarly, r_i denotes the range of sensor s_i. The SCAN algorithm consists of the following three steps:

1. Topology R is minimised by removing all redundant sensor ranges, i.e., sensor ranges which are fully covered by the combination of one or more other sensor ranges. The minimised topology is

[1] The sketch of an algorithm supporting sensing errors is given in [8]. We ignore it here as it falls outside the scope of this work.



(a) (b)

Figure 1: Example topologies with three (a) and nine (b) targets. In both examples, each sensor counts 3 targets, the maximum overlap degree is 3, and the estimated count computed by SCAN is $3\sqrt{3} \approx 5.2$, which leads to bound $[3, 9]$.

indicated by R', while the corresponding sensors and their readings are sets S' and C', respectively. Intuitively, no range can be deleted from R' without losing some coverage of the underlying space.

2. The maximum overlap degree (m) of R' is computed.
3. Degree m is then used to compute the estimated number of targets $\hat{t}$ and its corresponding bounds:

$$\hat{t} = \frac{\sum_{c_i \in C'} c_i}{\sqrt{m}}, \qquad \left[\frac{\hat{t}}{\sqrt{m}}, \hat{t}\sqrt{m}\right].$$

Note that $\hat{t}$ is the geometric mean of the bounds.

The two example topologies in Fig. 1 can help shed some light on how SCAN works and why the estimation bounds are defined in this manner. In both examples, there are three non-redundant sensors, the maximum overlap degree is 3 and each sensor counts three targets. However, in Fig. 1a, the total number of targets is 3, while, in Fig. 1b, it is 9. In both cases, SCAN computes the estimated number of targets $\hat{t} = 3\sqrt{3} \approx 5.2$ and its bounds $[3, 9]$. Observe that the sensor counts do not allow to distinguish the two cases. The lowest possible true count occurs when all the targets are over-counted (three times as the max overlap degree is 3), i.e., 3 as in Fig. 1a. The maximum true count arises when the readings are not over counted, i.e., 9 as in Fig. 1b. These are exactly the bounds computed by SCAN.

Discussion. The SCAN algorithm is computationally efficient and easy to deploy at a base station knowing the precise geometry of the sensors locations and ranges. However, the algorithm gives acceptable estimates (i.e., tight bounds) only when the maximum overlap degree is low. This follows from the definition of estimate bounds and their width: $s - \frac{s}{m}$, with $s = \sum_{c_i \in C'} c_i$. Another limitation of SCAN is that the presentation given by its authors fails to include a rigorous procedure to minimise topologies in two-dimensional space. This is a crucial issue as we will discuss extensively in the next section. Also, the algorithm disregards all the readings produced by redundant sensors, raising the question if this information could otherwise be used to obtain better estimates.

3 Static sensor topologies

In this section, we will formalise the topological aspects of the SCAN algorithm presented in Sect. 2. That is, we present an algorithm to represent step 1, by removing sensors from a given topology, such that

all sensors are needed for coverage. Furthermore, we present means to compute the maximum number of overlapping sensor ranges in such a topology, to cover step 2. To that end, we present the *Sensor Topology Logic* (STL) to reason about topologies created by overlapping sensor ranges. STL is a two-sorted first-order logic with equality and a single predicate between the sorts. The models of STL are abstractions of sensor range topologies, inspired by the formalisations of Euler diagrams [10]. We will use this logic to specify properties arising from two algorithms implementing the topological computations of SCAN in two dimensions. Furthermore, we prove these algorithms to be correct using Hoare logic [9]. However, we will not present fully formal Hoare logic proofs, but rather *proof outlines* [12]. Note, our models are qualitative, and hence invariant under many geometric deformations.

Program verification using proof outlines. A proof outline consists of the program to prove, annotated with assertions between the program statements.[2] These assertions are enclosed in curly braces and consist of properties in a specification language, which is STL in our case. An annotated statement $\{P\}s\{Q\}$ denotes that Q holds after the execution of s under the precondition P. That is, if P holds initially, Q holds after executing s. If two assertions P and Q directly follow each other in a proof outline, i.e., $\{P\}\{Q\}$, then we have to show that P implies Q. For conditional statements *if* P *then* ... *else* ... *end if*, the condition is added as an assertion to the beginning of the *if* branch, while its negation is the first annotation on the *else* branch. That is, we end up with the structure *if* P *then* $\{P\}$ ... *else* $\{\neg P\}$ ... *end if*. To verify a *while* loop, i.e., *while* Q *do* ... *end while*, we have to identify the *loop invariant* P. If we can show that P holds before entering the loop and after an execution of the loop, then we can also infer that P holds after leaving the loop. Then, the conjunction of the invariant P and the loop condition Q is the first assertion within the loop, while the conjunction $P \wedge \neg Q$ is the first assertion after the loop. That is, we have $\{P\}$ *while* Q *do* $\{P \wedge Q\}$ ... $\{P\}$ *end while* $\{P \wedge \neg Q\}$.

3.1 Syntax and semantics

The language of STL contains two sorts. These sorts are *sensors* $\mathfrak{S}$ and *zones* $\mathfrak{Z}$. The sort of sensors is countably infinite and the sort for zones is derived from this sort. In particular, $\mathfrak{Z}$ is the powerset of $\mathfrak{S}$. However, models for the logic will by definition be finite. In addition to Boolean operators and first-order quantification, STL formulas only contain two predicates: equality and the element relation $z \,\varepsilon\, ran(s)$, denoting that the zone z is contained in the sensor range of s. It can also be read as the element relation of set theory, i.e., "the zone z is an element of the range of sensor s".

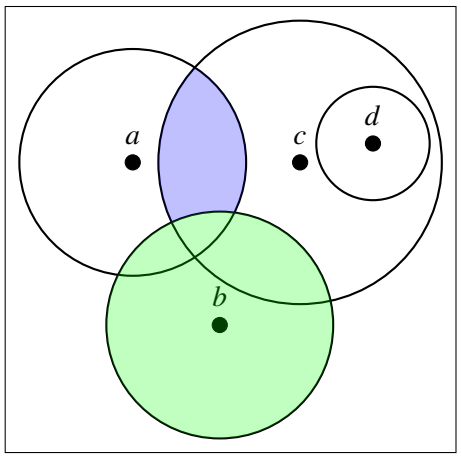
Definition 1 (Syntax of STL)**.** For a given $n \in \mathbb{N}$, the syntax of the logic is given by the following definitions. For each $1 \leq i \leq n$, s_i is a *sensor constant*. We denote the set of variables denoting zones or sensors by Var. If x is either a sensor constant or a sensor variable, we call x a *sensor term*. Formulas are given by the following EBNF:

$$\varphi ::= \bot \mid \theta = \theta \mid z \,\varepsilon\, ran(s) \mid \varphi \to \varphi \mid \forall x\colon \varphi$$

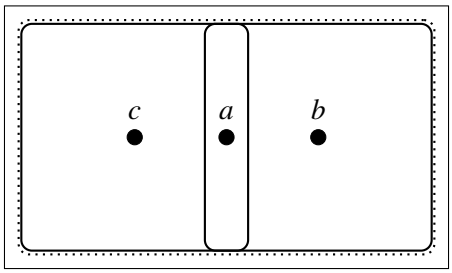
where θ are terms of the same sort, i.e., either zone variables or sensor terms, x is a variable, z is a zone variable and s is a sensor term. The other Boolean connectives $(\wedge, \vee, \ldots)$ and the existential quantifier are given by the usual abbreviations.

Models of STL consist of a finite set of sensors S, a set of zones Z defining the topology, and a function σ, where σ associates a set of zones to each sensor, to model its range. Formally, we have the following definition.

[2] For more in-depth discussions, see Hoare's original contribution [9] or Owicki and Gries' extension to parallel programs [12].



(a) Example of a Formal Topology



(b) Example of Ambiguous Irreducible Models

Figure 2: Examples of Topologies

Definition 2 (Static Topology Model)**.** A *static topology model* is a structure $\mathscr{M} = (S, Z, \sigma, \mathbf{s_1}, \ldots, \mathbf{s_n})$, where S is a set of sensors of sort $\mathfrak{S}$, $Z \subseteq \mathscr{P}(S)$ with $\emptyset \notin Z$ and for all $\mathbf{s} \in S$, there is at least one $z \in Z$ such that $\mathbf{s} \in z$. Furthermore, $\sigma\colon S \to \mathscr{P}(Z)$ is a function of arity 1, and $\mathbf{s_1}$ to $\mathbf{s_n}$ are enumerating S. For all $\mathbf{s_i}$, σ is defined by $\sigma(\mathbf{s_i}) = \{z \mid \mathbf{s_i} \in z \wedge z \in Z\}$. We will also write $\mathbf{s} \in \mathscr{M}$ for $\mathbf{s} \in S$. The set of all static topology models is denoted by $\mathbb{S}$. In the following, we will often omit the distinguished elements $\mathbf{s_i}$ from models. We denote the *empty model*, i.e., the model where $S = \emptyset$, simply by $\emptyset$. The *size* of a model $\mathscr{M} = (S, Z, \sigma)$, denoted by $|\mathscr{M}|$, is the number of sensors within $\mathscr{M}$, i.e., $|\mathscr{M}| = |S|$. Furthermore, we will use the notation $S_{\mathscr{M}}$, $Z_{\mathscr{M}}$, and $\sigma_{\mathscr{M}}$ to refer to the respective elements of $\mathscr{M}$.

Observe that each sensor constant of the syntax is associated with exactly one element of the sensor domain of a model. While we denote the syntactic sensor constant with letters in italics, e.g., s, t, a, ..., we use bold letters for the semantic sensors $\mathbf{s}$, $\mathbf{t}$, $\mathbf{a}$, respectively.

Consider the topology depicted in Fig. 2a, consisting of four sensors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$ and their respective sensor ranges. Within its formalisation $\mathscr{M}$, the intersection of $\mathbf{a}$ and $\mathbf{c}$ (coloured blue in the figure) is formalised by a zone $\{\mathbf{a}, \mathbf{c}\} \in Z$. The sensor range of $\mathbf{b}$ (coloured green) consists of all zones containing $\mathbf{b}$. It is given by the value of the function σ, i.e., $\sigma(\mathbf{b}) = \{\{\mathbf{b}\}, \{\mathbf{a}, \mathbf{b}\}, \{\mathbf{b}, \mathbf{c}\}, \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}\}$. Note that for a non-empty model, the definition of σ and Z ensures that $\sigma(\mathbf{s}) \neq \emptyset$ for all $\mathbf{s} \in S$. Furthermore, the empty model is uniquely defined as $(\emptyset, \emptyset, \emptyset)$.

Definition 3 (Semantics)**.** A *valuation* for a given model $\mathscr{M} = (S, Z, \sigma)$ is a function $\nu\colon \mathrm{Var} \to S \cup Z$, where $\nu(s) = \mathbf{s}$ and which respects the sorts of terms. The semantics of a formula with respect to a model $\mathscr{M}$ and a valuation ν is given by the following inductive definition.

$$
\begin{array}{ll}
\mathscr{M}, \nu \not\models \bot & \text{for all } \mathscr{M} \text{ and } \nu \\
\mathscr{M}, \nu \models \theta_1 = \theta_2 & \iff \nu(\theta_1) = \nu(\theta_2), \text{ where } \theta_1 \text{ and } \theta_2 \text{ are of the same sort} \\
\mathscr{M}, \nu \models z \,\varepsilon\, ran(s) & \iff \nu(z) \in \sigma(\nu(s)), \text{ where } z \text{ is a term of sort } \mathfrak{Z} \text{ and } s \text{ of } \mathfrak{S} \\
\mathscr{M}, \nu \models \varphi_1 \to \varphi_2 & \iff \mathscr{M}, \nu \models \varphi_1 \text{ implies } \mathscr{M}, \nu \models \varphi_2 \\
\mathscr{M}, \nu \models \forall x\colon \varphi & \iff \text{for all correctly sorted } \alpha \in \mathscr{M}, \nu[x \mapsto \alpha] \models \varphi
\end{array}
$$

A formula φ is *valid in a model* $\mathscr{M}$, denoted by $\mathscr{M} \models \varphi$, if $\mathscr{M}, \nu \models \varphi$ for all valuations ν. Similarly, a formula φ is *valid*, written $\models \varphi$, if it is valid in all models.

Using STL, we can express relations between different sensor ranges. In particular, we can mimic the subset relation, and we overload the equality sign to cover full sensor ranges as well.

$$ran(s) \leq ran(t) \equiv \forall x\colon x \,\varepsilon\, ran(s) \rightarrow x \,\varepsilon\, ran(t)$$
$$ran(s) = ran(t) \equiv \forall z\colon z \,\varepsilon\, ran(s) \leftrightarrow z \,\varepsilon\, ran(t)$$

Furthermore, we will employ several additional abbreviations. The formula $dj(s,t)$ expresses that the ranges of s and t are disjoint. Observe that this relation is symmetric. Indeed, it is the negation of $co(s,t)$, which denotes that the ranges of s and t have a common element. The abbreviation $ov(s,t)$ states that the ranges of s and t have a common element, but neither is a subset of the other. That is, the ranges of s and t are properly overlapping. Finally, $psubs(s,t)$ denotes that the sensor range of s is a proper subset of t's sensor range.

$$\begin{aligned} dj(s,t) &\equiv \forall x\colon x \,\varepsilon\, ran(s) \rightarrow \neg x \,\varepsilon\, ran(t) \\ co(s,t) &\equiv \exists x\colon x \,\varepsilon\, ran(s) \wedge x \,\varepsilon\, ran(t) \\ ov(s,t) &\equiv co(s,t) \wedge \neg(ran(s) \leq ran(t)) \wedge \neg(ran(t) \leq ran(s)) \\ psubs(s,t) &\equiv (ran(s) \leq ran(t)) \wedge \neg(ran(t) \leq ran(s)) \end{aligned}$$

Consider the topology of Fig. 2a and its formalisation $\mathscr{M}$. The sensor ranges in $\mathscr{M}$ are given by

$$\sigma(\mathbf{a}) = \{\{\mathbf{a}\},\{\mathbf{a},\mathbf{b}\},\{\mathbf{a},\mathbf{c}\},\{\mathbf{a},\mathbf{b},\mathbf{c}\}\} \qquad \sigma(\mathbf{b}) = \{\{\mathbf{b}\},\{\mathbf{a},\mathbf{b}\},\{\mathbf{b},\mathbf{c}\},\{\mathbf{a},\mathbf{b},\mathbf{c}\}\}$$
$$\sigma(\mathbf{c}) = \{\{\mathbf{c}\},\{\mathbf{a},\mathbf{c}\},\{\mathbf{b},\mathbf{c}\},\{\mathbf{c},\mathbf{d}\},\{\mathbf{a},\mathbf{b},\mathbf{c}\}\} \qquad \sigma(\mathbf{d}) = \{\{\mathbf{c},\mathbf{d}\}\}$$

That is, we have, for example,

$$\mathscr{M} \models \forall z\colon z \,\varepsilon\, ran(d) \rightarrow z \,\varepsilon\, ran(c)\ , \qquad \mathscr{M} \models \exists z\colon z \,\varepsilon\, ran(c) \wedge \neg z \,\varepsilon\, ran(d)\ ,$$
$$\mathscr{M} \models psubs(d,c)\ , \qquad \mathscr{M} \models dj(a,d)\ .$$

3.2 Identifying redundant sensors

In this section, we present the algorithm to formalise step 1 of the SCAN algorithm. The following formula can be used to identify sensors whose sensor range is covered by a union of other sensor ranges. That is, such a sensor can be removed from a model without impact on coverage.

$$red(s) \equiv \forall x\colon x \,\varepsilon\, ran(s) \rightarrow \exists y\colon y \neq s \wedge x \,\varepsilon\, ran(y)$$

Hence, a sensor is redundant if, and only if, all parts of space within its range are also covered by a different sensor. Using this formula, we can give the notion of an irreducible model.

Definition 4 (Irreducible Model)**.** A non-empty model $\mathscr{M}$ is *irreducible*, if it contains no redundant sensor. That is, $\mathscr{M} \models \forall s\colon \neg red(s)$. If $\mathscr{M} = (S,Z,\sigma)$ is a non-empty model and $\mathscr{M}^M = (S^M,Z^M,\sigma^M)$ is an irreducible model such that $S^M \subseteq S$, $Z^M = \{z^M \mid \exists z \in Z\colon z^M = z \cap S^M\}$, and σ^M is the function induced by S^M and Z^M, then we call $\mathscr{M}^M$ *an irreducible model of* $\mathscr{M}$.

Note that irreducible models of a given model are not unique. Consider, for example, the topology indicated in Fig. 2b, where the sensor range of $\mathbf{b}$ and $\mathbf{c}$ is denoted by solid rectangles, while the sensor range of $\mathbf{a}$ is drawn dotted. This topology is formalised by $\mathscr{M} = (\{\mathbf{a},\mathbf{b},\mathbf{c}\},Z,\sigma)$, where $Z = \{\{\mathbf{a},\mathbf{b}\},\{\mathbf{a},\mathbf{b},\mathbf{c}\},\{\mathbf{a},\mathbf{c}\}\}$, that is, $\sigma(\mathbf{a}) = \sigma(\mathbf{b}) \cup \sigma(\mathbf{c})$ and $\sigma(\mathbf{b}) \neq \sigma(\mathbf{c})$. Then both $\mathscr{M}_1 = (\{\mathbf{a}\},\{\{\mathbf{a}\}\},\sigma^1)$ and $\mathscr{M}_2 = (\{\mathbf{b},\mathbf{c}\},\{\{\mathbf{b}\},\{\mathbf{b},\mathbf{c}\},\{\mathbf{c}\}\},\sigma^2)$ are irreducible models of $\mathscr{M}$.

```
1: function REDUCE(𝓜)
2:     while 𝓜 ⊨ ∃t : red(t) do
3:         s ← chs({t | 𝓜 ⊨ red(t)})
4:         𝓜 ← 𝓜 \ {s}
5:     end while
6:     return 𝓜
7: end function
```

(a) Reduce a given model to an irreducible model

```
1: function MAXIMUM(𝓜)
2:     m ← |𝓜|
3:     while m > 1 do
4:         if 𝓜 ⊨ O^m then
5:             return m
6:         else
7:             m ← m − 1
8:         end if
9:     end while
10:    return m
11: end function
```

(b) Compute maximum number of overlaps

Figure 3: Topological Computations of SCAN

Definition 5 (Reduction)**.** Let $\mathscr{M} = (S, Z, \sigma)$ be a model, and $S' \subseteq S$ a proper subset of S. The *reduction of $\mathscr{M}$ by S'* is given by $\mathscr{M} \setminus S' = (S^R, Z^R, \sigma^R)$, where

1. $S^R = S \setminus S'$,
2. $Z^R = \{z^R \mid z \in Z \wedge z^R = z \setminus S'\}$, and
3. σ^R is the function induced by S^R and Z^R according to Def. 2.

We can prove that every non-empty sensor topology $\mathscr{M}$ contains an irreducible model. This by no means proves that this reduction is *unique*, since a suitable counterexample is already given in Fig. 2b. However, this lemma has to be carefully stated, since sensors that are redundant in $\mathscr{M}$, will not be redundant in irreducible models of $\mathscr{M}$. Hence, we can only state the existence of a subset of sensors that has to be removed to yield an irreducible model of $\mathscr{M}$.

Lemma 1. *Every non-empty model contains a non-empty irreducible model.*

$$\mathscr{M} \neq \emptyset \iff \exists R \subseteq \{\mathbf{c} \mid \mathscr{M} \models red(c)\} : \mathscr{M} \setminus R \neq \emptyset \text{ and } \mathscr{M} \setminus R \models \forall s : \neg red(s)$$

Proof. The direction from right to left is immediate, since $\mathscr{M}$ contains at least as many sensors as $\mathscr{M} \setminus R$ for all possible sets R. Now let $\mathscr{M} \neq \emptyset$ and proceed by induction on the number n of redundant sensors in $\mathscr{M}$. That is $n = |\{\mathbf{c} \mid \mathscr{M} \models red(c)\}|$. If $n = 0$, then $\mathscr{M}$ is irreducible and we can choose $R = \emptyset$ to finish the base case. For the induction step, let the lemma be true for all $n' \leq n$ and let $|\{\mathbf{c} \mid \mathscr{M} \models red(c)\}| = n+1$. Choose one $\mathbf{d} \in \{\mathbf{c} \mid \mathscr{M} \models red(c)\}$. Then $|\{\mathbf{c} \mid \mathscr{M} \setminus \{\mathbf{d}\} \models red(c)\}| < |\{\mathbf{c} \mid \mathscr{M} \models red(c)\}|$. Hence, by the induction hypothesis, there is an $R \subseteq \{\mathbf{c} \mid \mathscr{M} \setminus \{\mathbf{d}\} \models red(c)\}$ such that $(\mathscr{M} \setminus \{\mathbf{d}\}) \setminus R \neq \emptyset$ and $(\mathscr{M} \setminus \{\mathbf{d}\})) \setminus R \models \forall s : \neg red(s)$. Now let $R' = R \cup \{\mathbf{d}\}$. Then we have $\mathscr{M} \setminus R' \neq \emptyset$ and $\mathscr{M} \setminus R' \models \forall s : \neg red(s)$ and we are done. □

Figure 3a shows the algorithm allowing us to reduce a given model to an irreducible model covering the same space, thus representing step 1 of SCAN. The irreducible model returned by the algorithm is not unique, since the algorithm non-deterministically chooses a redundant sensor to remove from the model (denoted by the function *chs*) . Since the algorithm removes one sensor at a time, we have to ensure that the reduction of a model by a redundant sensor does again yield a non-empty model.

Lemma 2. *Let* $\mathbf{d}$ *be a redundant sensor in $\mathscr{M}$, i.e., $\mathscr{M} \models red(d)$. Then $\mathscr{M} \neq \emptyset \Rightarrow \mathscr{M} \setminus \{\mathbf{d}\} \neq \emptyset$.*

1: PRE:$\{\mathscr{M} \neq \emptyset\}$
2: **function** REDUCE($\mathscr{M}$)
3: $\{\mathscr{M} \neq \emptyset\}$
4: **while** $\mathscr{M} \models \exists t\colon red(t)$ **do**
5: $\{\mathscr{M} \models \exists t\colon red(t)) \wedge \mathscr{M} \neq \emptyset\}$
6: $\{\mathscr{M} \setminus \{chs(\{\mathbf{t} \mid \mathscr{M} \models red(t)\})\} \neq \emptyset$ (Lemma 2)$\}$
7: $s \leftarrow chs(\{\mathbf{t} \mid \mathscr{M} \models red(t)\})$
8: $\{\mathscr{M} \setminus \{s\} \neq \emptyset\}$
9: $\mathscr{M} \leftarrow \mathscr{M} \setminus \{s\}$
10: $\{\mathscr{M} \neq \emptyset\}$
11: **end while**
12: $\{\neg(\mathscr{M} \models \exists t\colon red(t)) \wedge \mathscr{M} \neq \emptyset\}$
13: $\{\mathscr{M} \models \forall t\colon \neg red(t) \wedge \mathscr{M} \neq \emptyset\}$
14: **return** $\mathscr{M}$
15: **end function**

(a) Proof Outline for Alg. 3a

1: PRE: $\{\mathscr{M} \neq \emptyset\}$
2: **function** MAXIMUM($\mathscr{M}$)
3: $\{|\mathscr{M}| \geq r_{\mathscr{M}} \wedge r_{\mathscr{M}} \geq 1\}$
4: $m \leftarrow |\mathscr{M}|$
5: $\{m \geq r_{\mathscr{M}}\}$
6: **while** $m > 1$ **do**
7: $\{m > 1 \wedge m \geq r_{\mathscr{M}}\}$
8: **if** $\mathscr{M} \models O^m$ **then**
9: $\{m \geq r_{\mathscr{M}} \wedge \mathscr{M} \models O^m\}$
10: $\{m = r_{\mathscr{M}}$ (Fact 1)$\}$
11: **return** m
12: **else**
13: $\{m \geq r_{\mathscr{M}} \wedge \mathscr{M} \not\models O^m\}$
14: $\{m - 1 \geq r_{\mathscr{M}}$ (Fact 1)$\}$
15: $m \leftarrow m - 1$
16: $\{m \geq r_{\mathscr{M}}\}$
17: **end if**
18: **end while**
19: $\{m \leq 1 \wedge m \geq r_{\mathscr{M}}\}$
20: $\{m = r_{\mathscr{M}}$ (Fact 1, Lemma 3)$\}$
21: **return** m
22: **end function**

(b) Proof Outline for Alg. 3b

Figure 4: Proof Outlines for Topological Computations of SCAN

Proof. Let $\mathscr{M} \neq \emptyset$ and $\mathscr{M} \models red(d)$. Hence, there is at least one sensor $\mathbf{s}$ different from $\mathbf{d}$ such that $\mathscr{M} \models d \neq s \wedge \exists x\colon x \,\varepsilon\, ran(d) \wedge x \,\varepsilon\, ran(s)$, since $\sigma(\mathbf{s}) \neq \emptyset$ for all $\mathbf{s}$. Due to the existence of $\mathbf{s}$, we get $\mathscr{M} \setminus \{\mathbf{d}\} \neq \emptyset$. □

With this lemma, we can provide a proof outline for the reduction algorithm, as shown in Fig. 4a. Note that the precondition of $\mathscr{M}$ being non-empty allows us to apply Lemma 2 while choosing a redundant sensor to remove. This also shows that the algorithm terminates, since otherwise it would have to remove redundant sensors infinitely often, which contradicts the finiteness of the models.

3.3 Compute maximum amount of overlap

In the following section, we show how to formalise step 2 of SCAN. To that end, we define the following formula for each m (bounded by the number of sensors in our model).

$$O^m \equiv \exists x, y_1, \ldots, y_m\colon \bigwedge_{i=1}^{m} x \,\varepsilon\, ran(y_i) \wedge \bigwedge_{i \neq j} y_i \neq y_j$$

That is, O^m is true, if, and only if there is a zone in the model which is covered by m distinct sensors. Observe that the sorting of terms (in particular, the application of *ran*) ensures that all the y_i are indeed sensors and not arbitrary elements of the space represented by Z. Then, we can show that every topology model contains a zone covered by at least one sensor.

Lemma 3. *Let $\mathscr{M} \neq \emptyset$. Then $\mathscr{M} \models O^1$.*

Proof. If $\mathscr{M} \neq \emptyset$, we know that there is at least one $\mathbf{s} \in S$ and consequently one $z \in \sigma(\mathbf{s})$. Observe that $O^1 \equiv \exists x,y\colon x \,\varepsilon\, ran(y)$. Now, let ν an arbitrary valuation. During the evaluation of the existential quantifier, we can choose the new value for x to be z and for y to be $\mathbf{s}$. Hence $\mathscr{M}, \nu \models O^1$. □

Figure 3b presents the formal algorithm of the computation of the maximal number of overlapping sensors. To prove it to be correct, we need some minor definitions. We let $r_{\mathscr{M}}$ be the maximum of the number of non-redundant overlapping sensors in the model $\mathscr{M}$, i.e.,

$$r_{\mathscr{M}} = \max\{m \mid \mathscr{M} \models O^m\} \ .$$

We want to show that our algorithm always returns $r_{\mathscr{M}}$. For a non-empty model $\mathscr{M} = (S, Z, \sigma)$, we have $r_{\mathscr{M}} \geq 1$ by Lemma 3. Furthermore, the definition of $r_{\mathscr{M}}$ establishes the following fact.

Fact 1. *The formula O^m is valid in a static topology model $\mathscr{M}$, if and only if m is at most the maximum amount of overlaps within $\mathscr{M}$. That is, $\mathscr{M} \models O^m \iff m \leq r_{\mathscr{M}}$.*

With these notions at hand, we are able to prove correctness of the algorithm in Fig. 3b (see Fig. 4b). We do not refer to topological properties in the proof outline, hence the specification language is just arithmetic. The loop invariant is that the variable m is greater or equal to the maximal amount of overlapping sensor ranges, i.e. $m \geq r_{\mathscr{M}}$. Observe that the algorithm terminates due to the finiteness of the models.

3.4 Non-destructive target estimation

In the previous sections, we formalised the topological aspects of the SCAN algorithm. However, the computation we presented is destructive, i.e., we remove sensors from a given model until we get a irreducible model. Hence, we removed information that could be used to tighten the estimation bounds computed by the algorithm. In this section, we show how this construction may be altered to preserve this information. To that end, instead of removing sensors, we will mark the sensors to be either necessary or unnecessary for the computation. This slightly complicates the algorithms, since we cannot simply identify the necessary sensors by finding zones that are only contained within a single sensor range, but we instead have to disregard the sensors already marked as unnecessary.

Hence, we extend the syntax and semantics with predicates *nec* and *unnec* to express, whether a sensor is considered (un-)necessary to cover the given space. To that end, we extend a model over the set of sensors S with two new sets $N \subseteq S$ and $U \subseteq S$, where $N \cap U = \emptyset$. Syntactically, both predicates are unary, and their semantics is given by

$$\begin{aligned} \mathscr{M}, \nu \models nec(s) &\iff \nu(\mathbf{s}) \in N\,, \\ \mathscr{M}, \nu \models unnec(s) &\iff \nu(\mathbf{s}) \in U\,. \end{aligned}$$

The predicates *nec* and *unnec* allow us during the algorithms to categorise sensors used for computing the estimation of targets. Consider the example given in Sect. 3.2 for irreducible models. If we mark **a** as necessary, but do not remove **b** and **c**, **a** would still satisfy the redundancy formula. We rectify this situation by relaxing our notion of redundancy to capture *possible redundancy*.

$$\begin{aligned} posred(s) &\equiv \forall x\colon x \,\varepsilon\, ran(s) \rightarrow \exists y\colon y \neq s \wedge \neg unnec(y) \wedge x \,\varepsilon\, ran(y) \\ cover &\equiv \forall z \exists s\colon \neg unnec(s) \wedge z \,\varepsilon\, ran(s) \end{aligned}$$

1: **function** REDUCE($\mathscr{M}$)
2: **while** $\mathscr{M} \models \exists s\colon posred(s)$ **do**
3: $s \leftarrow chs(\{\mathbf{s} \mid \mathscr{M} \models posred(s)\})$
4: $\mathscr{M} \leftarrow (S_{\mathscr{M}}, Z_{\mathscr{M}}, N_{\mathscr{M}}, U_{\mathscr{M}} \cup \{s\}, \sigma_{\mathscr{M}})$
5: **end while**
6: $\mathscr{M} \leftarrow (S_{\mathscr{M}}, Z_{\mathscr{M}}, S_{\mathscr{M}} \setminus U_{\mathscr{M}}, U_{\mathscr{M}}, \sigma_{\mathscr{M}})$
7: **return** $\mathscr{M}$
8: **end function**

Figure 5: Non-Destructive Reduction

In contrast to $red(s)$, the formula $posred(s)$ requires that the covering sensors must not be included in the set U, i.e., they may not be deemed unnecessary for full coverage. Of course, we have to modify the definition of irreducible models as well, since we will keep redundant sensors in the models.

Definition 6 (Irreducible Model (2))**.** Let $\mathscr{M} = (S, Z, N, U, \sigma)$ be a non-empty static topology model. We call $\mathscr{M}$ an *irreducible model*, if and only if it satisfies the following formula

$$(\forall x \exists y\colon x \,\varepsilon\, ran(y) \wedge nec(y)) \wedge (\forall s\colon nec(s) \vee unnec(s)) \wedge (\forall s\colon posred(s) \rightarrow unnec(s)).$$

That is, every zone in $\mathscr{M}$ is covered by at least one sensor marked as necessary, every sensor is either necessary or unnecessary, and every possibly redundant sensor is indeed marked as unnecessary.

Figure 5 shows the algorithm for non-destructive reduction. It is similar to the destructive algorithm, but instead of completely removing the sensors deemed unnecessary, it just marks them appropriately. Finally, all remaining sensors are marked as necessary. While we do not show a formal proof of termination, observe that in each step, a new sensor is marked as unnecessary. Since the model contains only finitely many sensors, the algorithm has to terminate.

The corresponding proof outline is presented in Fig. 6. The loop invariant is the formula *cover*, i.e., at each part of the loop, all zones in the model are covered by at least one sensor not yet marked as unnecessary. Most of the proof steps are direct application of the Hoare proof rules. However, there are three steps that are not immediate, First, we have to prove that the input of the algorithm, a model without any marked sensors, satisfies the loop invariant. This is shown in Lemma 4.

Lemma 4. *Let $\mathscr{M}$ be a non-empty model with $N_{\mathscr{M}} = U_{\mathscr{M}} = \emptyset$. Then $\mathscr{M} \models cover$.*

Proof. Since $U_{\mathscr{M}} = \emptyset$, no sensor is marked as unnecessary. Furthermore, due to our general condition on topology models, each zone is covered by at least one sensor. All in all, each zone is covered by at least one sensor, which is not marked as unnecessary. □

The next lemma is used to prove the step from line 5 to line 6. That is, if the loop condition and the invariant hold for a model, then we can choose a possibly redundant sensor s that has not yet been marked, mark it as unnecessary, and the resulting model still satisfies the invariant *cover*.

Lemma 5. *Let $\mathscr{M}$ be a model with $\mathscr{M} \models \forall z \exists s\colon \neg unnec(s) \wedge z \,\varepsilon\, ran(s)$ and s be a sensor constant such that $\mathscr{M} \models \neg unnec(s) \wedge posred(s)$. Furthermore, let $\mathscr{M}' = (S_{\mathscr{M}}, Z_{\mathscr{M}}, N_{\mathscr{M}}, U_{\mathscr{M}} \cup \{\mathbf{s}\}, \sigma_{\mathscr{M}})$. Then $\mathscr{M}' \models \forall z \exists s\colon \neg unnec(s) \wedge z \,\varepsilon\, ran(s)$.*

Proof. Observe that the zones and their containment relations in the ranges does not change between $\mathscr{M}$ and $\mathscr{M}'$. For all z with $\mathscr{M}' \models \neg z \,\varepsilon\, ran(s)$, we have the result immediately, by the condition on

1: PRE:$\{\mathscr{M} \neq \emptyset \wedge N_{\mathscr{M}} = U_{\mathscr{M}} = \emptyset\}$
2: **function** REDUCE($\mathscr{M}$)
3: $\{\mathscr{M} \models cover$ (Lemma 4)$\}$
4: **while** $\mathscr{M} \models \exists s\colon \neg unnec(s) \wedge posred(s)$ **do**
5: $\{\mathscr{M} \models cover$ and $\mathscr{M} \models \exists s\colon \neg unnec(s) \wedge posred(s)\}$
6: $\{(S_{\mathscr{M}}, Z_{\mathscr{M}}, N_{\mathscr{M}}, U_{\mathscr{M}} \cup \{chs(\{\mathbf{s} \mid \mathscr{M} \models \neg unnec(s) \wedge posred(s)\}), \sigma_{\mathscr{M}}) \models cover\}$ (Lem. 5)$\}$
7: $s \leftarrow chs(\{\mathbf{s} \mid \mathscr{M} \models \neg unnec(s) \wedge posred(s)\})$
8: $\{(S_{\mathscr{M}}, Z_{\mathscr{M}}, N_{\mathscr{M}}, U_{\mathscr{M}} \cup \{s\}, \sigma_{\mathscr{M}}) \models cover\}$
9: $\mathscr{M} \leftarrow (S_{\mathscr{M}}, Z_{\mathscr{M}}, N_{\mathscr{M}}, U_{\mathscr{M}} \cup \{s\}, \sigma_{\mathscr{M}})$
10: $\{\mathscr{M} \models cover\}$
11: **end while**
12: $\{\mathscr{M} \models cover$ and $\mathscr{M} \models \forall s\colon unnec(s) \vee \neg posred(s)\}$
13: $\{(S_{\mathscr{M}}, Z_{\mathscr{M}}, S_{\mathscr{M}} \setminus U_{\mathscr{M}}, U_{\mathscr{M}}, \sigma_{\mathscr{M}})$ is irreducible. (Lemma 6)$\}$
14: $\mathscr{M} \leftarrow (S_{\mathscr{M}}, Z_{\mathscr{M}}, S_{\mathscr{M}} \setminus U_{\mathscr{M}}, U_{\mathscr{M}}, \sigma_{\mathscr{M}})$
15: $\{\mathscr{M}$ is irreducible.$\}$
16: **return** $\mathscr{M}$
17: **end function**

Figure 6: Proof Outline for Alg. 5.

$\mathscr{M}$. So let z be a zone with $\mathscr{M}' \models z \,\varepsilon\, ran(s)$. By $\mathscr{M} \models posred(s)$, we know that there is a sensor t different from s with $\mathscr{M} \models \neg unnec(t) \wedge z \,\varepsilon\, ran(t)$. Since in $\mathscr{M}'$ we still have $\mathscr{M}' \models \neg unnec(t)$, we get $\mathscr{M}' \models \exists t\colon s \neq t \wedge \neg unnec(t) \wedge z \,\varepsilon\, ran(s)$. Since z was arbitrary, the lemma follows. □

The last lemma of this section refers to the final steps in the algorithm, i.e., the relation between line 12 and 13. It shows that if we have a model that satisfies the loop invariant, and where all sensors are either marked as unnecessary or are not possibly redundant, then the model created by marking all remaining sensors as necessary is irreducible. That is, this lemma states that the model returned by the algorithm is always irreducible.

Lemma 6. *Let $\mathscr{M}$ be model such that $\mathscr{M} \models cover$ and $\mathscr{M} \models \forall s\colon unnec(s) \vee \neg posred(s)$. Furthermore, let $\mathscr{M}' = (S_{\mathscr{M}}, Z_{\mathscr{M}}, S_{\mathscr{M}} \setminus U_{\mathscr{M}}, U_{\mathscr{M}}, \sigma_{\mathscr{M}})$. Then $\mathscr{M}'$ is irreducible, i.e.,*

$$\mathscr{M}' \models \forall x \exists y\colon x \,\varepsilon\, ran(y) \wedge nec(y) \tag{1}$$
$$\mathscr{M}' \models \forall s\colon nec(s) \vee unnec(s) \tag{2}$$
$$\mathscr{M}' \models \forall s\colon posred(s) \rightarrow unnec(s). \tag{3}$$

Proof. Property (2) holds by construction of $\mathscr{M}'$. Furthermore, we get property (3) by the assumption on $\mathscr{M}$, and the fact that $\mathscr{M}'$ contains the same set of unnecessary sensors as $\mathscr{M}$. Finally, property (1) holds, since $\mathscr{M} \models cover$, and for all sensors s with $\mathscr{M} \models \neg unnec(s)$, we have $\mathscr{M}' \models nec(s)$. □

For the counterpart to the algorithm shown in Fig. 3b, we can amend the formulas to only use sensors deemed necessary. The algorithm itself does not need to be changed.

$$O^m \equiv \exists x, y_1, \ldots, y_m\colon \bigwedge_{i=1}^{m} x \,\varepsilon\, ran(y_i) \wedge \bigwedge_{i \neq j} y_i \neq y_j \wedge \bigwedge_{i=1}^{m} nec(y_i)$$

4 Dynamical topology models

The models defined in Sect. 3 define static topologies of sensor nodes. However, in reality, topological relations of the sensor ranges may change over time, for example due to environmental reasons like rain or fog, or moving obstacles blocking the sensors, or because the sensors are mobile, e.g., attached to drones or autonomous robots. This may result in previously unnecessary sensors becoming necessary and vice versa. In this section, we present how we can extend the formal models to reason about temporal changes in the topologies, by temporalising our static logic with respect to *full computation tree logic* (CTL*) [6]. Since our models are qualitative in nature, we only capture purely qualitative changes within the topology. Egenhofer and Al-Taha [5] identified the possible changes in qualitative relationships for spatial regions, i.e., regions that are non-empty, convex connected subsets of $\mathbb{R}^2$. This matches our view of the sensor ranges. The full set of relations between regions is shown in Tab. 1.

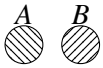
disjoint		contains	
inside		equal	
meet		covers	
coveredBy		overlap	

Table 1: The eight possible relations between spatial regions A and B [5]

However, in contrast to their work, we have no means of identifying the boundary of a spatial region. Furthermore, we assume that targets are only countable, if they are fully within the sensor range, and not only on the boundary. As a consequence, the differences between several of these relations are not of interest to us. For example, the difference between *meet* and *disjoint* is only whether the boundaries intersect. Similarly, we cannot distinguish between *covers* and *contains*, as well as *coveredBy* and *inside*. From each of these three pairs, we keep the less specific relation. Hence we end up with the five relations: *disjoint*, *equal*, *overlap*, *inside*, and *contains*. We follow Egenhofer and Al-Taha's approach of identifying the pairs of relations with the least topological distance, while ignoring the relations concerning the boundaries. This results in the graph of possible qualitative changes for our model as shown in Fig. 7. If two regions move relative to each other, the relation between them can only change as indicated by the graph. For example, if two initially disjoint regions move, the relation between them cannot immediately change such that one is a subregion of the other, but they have to partially overlap first.

4.1 Syntax and semantics

We extend the notion of a model along the lines of Finger and Gabbay [7]. That is, we use a standard possible-world semantics, but associate a static model as defined in the previous section to each world. The worlds are connected by directed transitions, to form a model of *branching time*. Hence, at each world, several futures are possible. While our definition of models does not respect the relational changes identified in in Fig. 7, we will restrict the connections between worlds to adhere to the qualitative changes given by requiring certain axioms to hold.

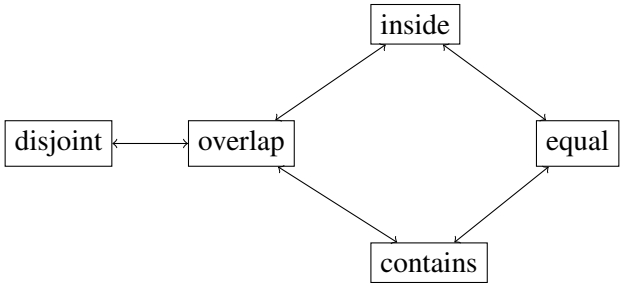


Figure 7: Possible qualitative changes of two sensor ranges

We only define the dynamic models with respect to the topologies of the sensors. That is, we associate a point in time with a class of models, which all possess the same topology, but where different sensors are marked as necessary or unnecessary. Formally, we say that two models $\mathscr{M}_1 = (S_1, Z_1, N_1, U_1, \sigma_1)$ and $\mathscr{M}_2 = (S_2, Z_2, N_2, U_2, \sigma_2)$ are *topologically equivalent*, written $\mathscr{M}_1 \sim \mathscr{M}_2$, if and only if $S_1 = S_2$, $Z_1 = Z_2$, and $\forall \mathbf{s}\colon \sigma_1(\mathbf{s}) = \sigma_2(\mathbf{s})$. We denote the equivalence class of a model $\mathscr{M}$ by $[\mathscr{M}]$ and the set of these equivalence classes by $\mathbb{S}_{/\sim}$.

Definition 7 (Dynamic Topology Model)**.** A *dynamic topology model* is a structure $\mathscr{D} = (W, \Rightarrow, f)$, where W is a non-empty set of *worlds*, $\Rightarrow \subseteq W \times W$ is the *accessibility relation* and $f\colon W \to \mathbb{S}_{/\sim}$ is a function mapping each world to an equivalence class of static topology models. Furthermore, we require that the members of the equivalence classes in the range of f are based on the same domain of sensors.

Definition 8 (Path)**.** Let $\mathscr{D}$ be a model according to Def. 7. A *path* π through $\mathscr{D}$ is an infinite sequence of worlds $w_0, w_1, \ldots$, such that $w_i \in \mathscr{D}$ and $w_i \Rightarrow w_{i+1}$ for all $i \in \mathbb{N}$. Given a path $\pi = w_0, w_1, \ldots$, we will also denote w_i by π^i. We write the *set of all paths in* $\mathscr{D}$ as $Paths(\mathscr{D})$. Furthermore, the set of all paths in $\mathscr{D}$ starting at w is denoted by $Paths_w(\mathscr{D}) = \{\pi \mid \pi \in Paths(\mathscr{D}) \wedge \pi^0 = w\}$.

The syntax of *Dynamic STL* is based on CTL*, but instead of propositional atoms, we use static topology formulas as defined in the previous section.

Definition 9 (Syntax of Dynamic STL)**.** The syntax of Dynamic STL is given by the following EBNF.

$$\chi ::= \varphi_s \mid \neg\chi \mid \chi \wedge \chi \mid \mathbf{A}\psi$$
$$\psi ::= \chi \mid \neg\psi \mid \psi \wedge \psi \mid \mathbf{X}\chi \mid \chi \mathbf{U} \chi$$

where φ_s is a static topology formula, which does not use necessity predicates. We call formulas of the form χ *state formulas* and formulas of type ψ *path formulas*.

The other operators, i.e., the Boolean operators like disjunction, implication, the temporal operators **G** and **F** as well as the remaining path operator **E** can be defined as abbreviations as usual [6]. Observe that the semantics of purely topological formulas is equivalent for all members of a topological equivalence class. Hence we silently lift satisfaction of these formulas to topological equivalence classes

Definition 10 (Semantics)**.** Let $\mathscr{D}$ be a dynamic topology model and $w \in \mathscr{D}$. The semantics of a formula with respect to $\mathscr{D}$ and a valuation ν is given by the following inductive definition. The Boolean operators are defined as usual, and hence we omit them.

$$\begin{aligned}
&\mathscr{D}, w, \nu \models \varphi_s && \iff f(w), \nu \models \varphi_s \\
&\mathscr{D}, w, \nu \models \mathbf{A}\psi && \iff \forall \pi \in Paths_w(\mathscr{D})\colon \mathscr{D}, \pi, \nu \models \psi \\
&\mathscr{D}, \pi, \nu \models \mathbf{X}\chi && \iff \mathscr{D}, \pi^1, \nu \models \chi \\
&\mathscr{D}, \pi, \nu \models \chi_1 \mathbf{U} \chi_2 && \iff \exists k \in \mathbb{N}\colon \mathscr{D}, \pi^k, \nu \models \chi_2 \text{ and } \forall i < k\colon \mathscr{D}, \pi^i, \nu \models \chi_1
\end{aligned}$$

4.2 Axiomatising semantical conditions

Recall the abbreviations from Sect. 3.1 for expressing proper overlaps and subsets. With these abbreviations at hand, we can represent the changes allowed by Fig. 7 by suitable axioms. Since the conditions only concern single transitions, the axioms are of the form $\varphi_1 \to \mathbf{AX}\varphi_2$. That is, they describe how a given state (identified by φ_1) may evolve during one transition. The left-hand side of each axiom directly corresponds to a state in Fig. 7. The right-hand sides reflect the outgoing edges of each state as succinct as possible. For example, in Axiom (4), the right-hand side ensures, that neither range is a subset of the other. This condition is satisfied, either if the ranges are disjoint or if they partially overlap, but in no other case.

$$dj(s,s') \to \mathbf{AX}(\neg ran(s) \leq ran(s') \wedge \neg ran(s') \leq ran(s)) \tag{4}$$

$$ov(s,s') \to \mathbf{AX}(\neg ran(s) \leq ran(s') \vee \neg ran(s') \leq ran(s)) \tag{5}$$

$$psubs(s,s') \to \mathbf{AX}\left(ran(s) \leq ran(s') \vee ov(s,s')\right) \tag{6}$$

$$ran(s) = ran(s') \to \mathbf{AX}(ran(s) \leq ran(s') \vee ran(s') \leq ran(s)) \tag{7}$$

The four axioms model the possible qualitative topological changes. E.g., axioms 4 denotes that two disjoint sensor ranges can either stay disjoint at the next step, or may overlap without one range covering the other. Observe that axiom 6 handles both cases, where one range is covered by the other (i.e., the *inside* and *contains* relations of Fig. 7), since these relations are inverse to each other.

5 Conclusion

We showed how the topological aspects of the SCAN target counting algorithm in two-dimensions can be formalised. To that end, we defined a notion of sensor models and a suitable first-order logic, dedicated to reason about specific parts of the topology, in particular, the intersections of sensor ranges. This allowed us to uncover non-determinism in the algorithm and specify how this may affect the end results. Namely, we showed that the notion of irreducible topologies is not unique and how this can have a negative impact on the precision of the bounds computed by the algorithm. We then extended our model to take dynamic topologies into account: first, we identified with a lattice in which ways a topology may change when a sensor moves; then, we amended the formalisation of the algorithm to accommodate these changes.

We believe that the logical formalisation presented in this paper will serve as a valuable tool for the rigorous study of the topology-dependent algorithms we often encounter in wireless sensor networks, ubiquitous systems, IoT, and mixed reality systems.

In future work, we intend to analyse the SCAN algorithm's numerical estimates and possible improvements. To that end, we will extend STL by associating each sensor range with its reading, i.e., the number of targets on its range. For example, we could model sensor readings with intervals $[a,b]$ to represent the upper and lower bounds of each reading. Then, instead of being discarded, redundant sensor readings could be used to refine the values of the non-redundant sensors used in the computations, potentially tightening the bounds of the SCAN algorithm's estimates. Another issue worth investigating is to analyse how miscounts in the individual readings perturb the final estimates.

A second strand of research will involve comparing different estimation schemes and establishing links with other formalisations of the target counting problem such as the approach based on Euler characteristic over simplicial complexes introduced by Baryshnikov and Ghrist [2]. Another formalism we will consider is bigraphs with sharing [14], a computational model with strong emphasis on spatial

properties of systems that has been used by one of the authors to formalise topological aspects of wireless networks [3].

References

[1] Marco Aiello, Ian Pratt-Hartmann & Johan van Benthem, editors (2007): *Handbook of Spatial Logics*. Springer, doi:10.1007/978-1-4020-5587-4.

[2] Yuliy Baryshnikov & Robert Ghrist (2009): *Target enumeration via Euler characteristic integrals*. *SIAM Journal on Applied Mathematics* 70(3), pp. 825–844, doi:10.1137/070687293.

[3] Muffy Calder & Michele Sevegnani (2014): *Modelling IEEE 802.11 CSMA/CA RTS/CTS with stochastic bigraphs with sharing*. *Formal Aspects of Computer Science* 26, pp. 537–561, doi:10.1007/s00165-012-0270-3.

[4] Vincenzo Ciancia, Diego Latella, Michele Loreti & Mieke Massink (2014): *Specifying and Verifying Properties of Space*. In: *Theoretical Computer Science: 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014*, Springer, pp. 222–235, doi:10.1007/978-3-662-44602-7˙18.

[5] Max J. Egenhofer & Khaled K. Al-Taha (1992): *Reasoning About Gradual Changes of Topological Relationships*. In: *Proceedings of the International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Springer, pp. 196–219, doi:10.1007/3-540-55966-3˙12.

[6] E. Allen Emerson & Joseph Y. Halpern (1986): *"Sometimes" and "not never" revisited: on branching versus linear time temporal logic*. *J. ACM* 33(1), pp. 151–178, doi:10.1145/4904.4999.

[7] Marcelo Finger & Dov M. Gabbay (1992): *Adding a temporal dimension to a logic system*. *Journal of Logic, Language and Information* 1(3), pp. 203–233, doi:10.1007/BF00156915.

[8] Sorabh Gandhi, Rajesh Kumar & Subhash Suri (2008): *Target Counting under Minimal Sensing: Complexity and Approximations*. In: *Algorithmic Aspects of Wireless Sensor Networks: Fourth International Workshop, ALGOSENSORS 2008, Reykjavik, Iceland, July 2008. Revised Selected Papers*, Springer, pp. 30–42, doi:10.1007/978-3-540-92862-1˙4.

[9] C. A. R. Hoare (1969): *An Axiomatic Basis for Computer Programming*. *Communications of the ACM* 12(10), pp. 576–580, doi:10.1145/363235.363259.

[10] John Howse, Fernando Molina, Sun-Joo Shin & John Taylor (2002): *On Diagram Tokens and Types*. In: *Diagrammatic Representation and Inference: Second International Conference, Diagrams 2002 Callaway Gardens, GA, USA*, Springer, pp. 146–160, doi:10.1007/3-540-46037-3˙18.

[11] Sven Linker & Martin Hilscher (2015): *Proof Theory of a Multi-Lane Spatial Logic*. *Logical Methods in Computer Science* 11(3), doi:10.2168/LMCS-11(3:4)2015.

[12] Susan Owicki & David Gries (1976): *An Axiomatic Proof Technique for Parallel Programs I*. *Acta Inf.* 6(4), pp. 319–340, doi:10.1007/BF00268134.

[13] David A. Randell, Zhan Cui & Anthony G. Cohn (1992): *A Spatial Logic based on Regions and Connection*. In B. Nebel, C. Rich & W. Swartout, editors: *International Conference on Principles of Knowledge Representation and Reasoning – KR 1992*, Morgan Kaufmann, pp. 165–176.

[14] Michele Sevegnani & Muffy Calder (2015): *Bigraphs with sharing*. *Theoretical Computer Science* 577, pp. 43 – 73, doi:10.1016/j.tcs.2015.02.011.

[15] Dengyuan Wu, Dechang Chen, Kai Xing & Xiuzhen Cheng (2012): *A statistical approach for target counting in sensor-based surveillance systems*. In: *INFOCOM, 2012 Proceedings IEEE*, pp. 226–234, doi:10.1109/INFCOM.2012.6195613.

[16] Dengyuan Wu, Bowu Zhang, Hongjuan Li & Xiuzhen Cheng (2014): *Target Counting in Wireless Sensor Networks*, pp. 235–269. Springer, doi:10.1007/978-3-642-40066-7˙7.