A One-Dimensional Sparse Space-Time Specification of the Generalized Railroad Crossing

Michael Gosnell, Bruce Mcmillin

Chapter 14

A ONE-DIMENSIONAL SPARSE SPACE-TIME SPECIFICATION OF THE GENERALIZED RAILROAD CROSSING

Michael Gosnell and Bruce McMillin

Abstract Modeling and reasoning about critical infrastructure systems is a complex endeavor. Various calculi and algebras have been crafted to help specify physical properties such as time and space, but these do not always translate well between physical entities and their conceptual specifications. Although real-world critical infrastructure systems involve components of both time and space, many existing specification methods focus most strongly on the temporal components, leaving spatial details largely ignored or forcing then to fit within the confines of the temporal specification. This paper presents a one-dimensional sparse space-time specification created using a spatial-temporal logic in which real-world constraints are incorporated in the logic using the *next* operator. The simplicity and utility of the spatial-temporal formalism is demonstrated by applying it to the generalized railroad crossing problem.

Keywords: Generalized railroad crossing, sparse space-time, assertion checking

1. Introduction

Real-world critical infrastructure systems are susceptible to errors, including hardware malfunctions and failures, software malfunctions and corruption, malicious attacks, and unknown and unseen failures. While many techniques exist for helping mitigate errors, critical infrastructure protection is based on the assumption that the correct operation of the systems of interest is known. Expressing the correct operating behavior of a system can take on many forms, depending on the types of error mitigating techniques and personal preferences.

System specifications can be formulated in a variety of ways, such as using calculi [10], temporal logic [9, 10], or automata or state transition systems [5, 6] that can be automatically verified with model checking. Expressing the

Table 1. RCC interval relationships.

Interval	Definition
$C(x, y)$	x connects with y
$DC(x, y)$	x is disconnected from y
$P(x, y)$	x is a part of y
$PP(x, y)$	x is a proper part of y
$EQ(x, y)$	x is equivalent to y
$O(x, y)$	x overlaps y
$DR(x, y)$	x is discrete from y
$PO(x, y)$	x partially overlaps y
$EC(x, y)$	x is externally connected with y
$TPP(x, y)$	x is a tangential proper part of y
$NTPP(x, y)$	x is a nontangential proper part of y

correctness of a critical infrastructure system is often challenging due to the specification requirements. This paper presents a new "sparse space-time" approach, which is designed to better encapsulate physical characteristics within the specification, allowing for more natural specifications of components in the critical infrastructure protection domain.

2. Background

This section discusses the region connection calculus (RCC), which provides qualitative spatial relationships that are used within the specification language. Also, it discusses spatial-temporal logics that help capture temporal aspects. Finally, the generalized railroad crossing (GRC) problem is presented along with high-level definitions of safety and liveness.

2.1 Region Connection Calculus

The region connection calculus (RCC) [11] is an extension of the mereological- and topological-based work of Clarke [2] to form an interval logic for dealing with space. This interval spatial work incorporates qualitative relationships similar to Allen's interval temporal logic [1]. The RCC spatial interval relationships are summarized in Table 1, where $EQ(x, y)$ replaces the original $x = y$ notation. Note that P, PP, TPP and $NTPP$ are not symmetric and, therefore, support inverses, which are denoted by appending $^{-1}$ as in $NTPP^{-1}$.

In addition to the general RCC, a smaller set of jointly exhaustive pairwise disjoint relations are provided. Figure 1 shows these base relations along with the potential transitions between relations. These eight relationships form the basis of the region connection calculus known as RCC-8.

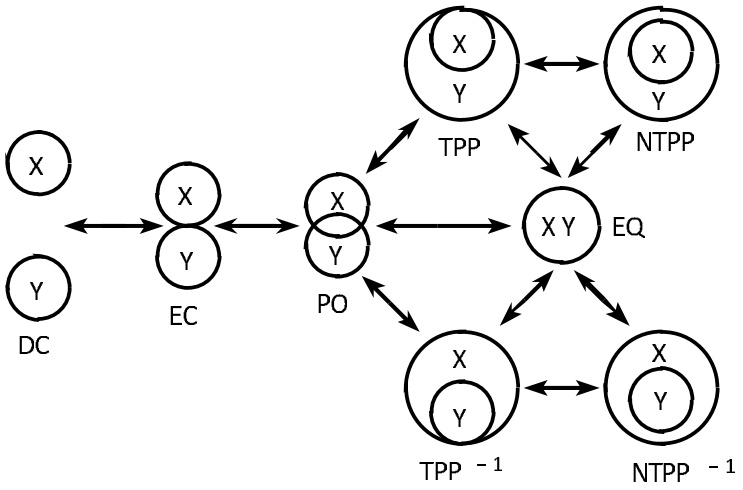


Figure 1. RCC-8 base relations and potential transitions.

2.2 Spatial-Temporal Logics

A spatial-temporal logic (STL) is the combination of a spatial logic with a temporal logic [7]. More specifically, an STL incorporates the expressiveness of the spatial and temporal logics, as well as the interactions between the spatial and temporal components as allowed by the STL. There are some standard characteristics of STLs, e.g., an STL should be able to specify spatial propositions in relation to time. However, the full specification of principles, which is unique to each particular STL, dictates how the spatial-temporal predicates extend individual spatial and temporal propositions and how truth values change over time. For example, the assertion:

$$NTPP(Computer\ Science, Campus) \Rightarrow \bigcirc\, NTPP(Computer\ Science, Campus)$$

states that if the Computer Science building is a (nontangential proper) part of the campus, it will remain so "at the next state" (where "state" might be a time, system state, world, etc.).

Multiple combinations of spatial and temporal logics are presented in [12], where RCC-8 is used as the basis for spatial reasoning. In this paper, STL specifications take the RCC-8 form with branching temporal logic as fits with STL logic $\mathcal{ST}_2$ in [12]. Because the focus is on aspects of specification languages as they relate to runtime assertion checking (and not model checking where issues such as decidability are of importance), additional technical aspects of STL are not included.

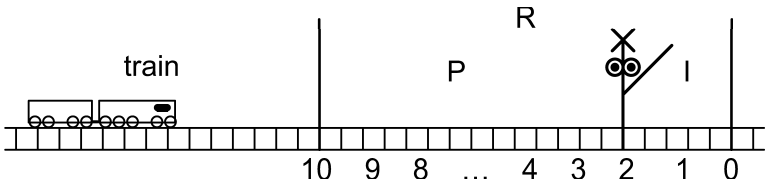


Figure 2. Generalized railroad crossing example.

2.3 Generalized Railroad Crossing

A generalized railroad crossing (GRC) problem is attractive as a basic scenario in a multitude of application domains. A railway can be interpreted (with some basic constraints) primarily as a one-dimensional domain, which reduces the complexity of having to conduct a multidimensional analysis. The railway is a component of the larger transportation infrastructure, and the model can be extended to vehicular, airspace and littoral domains. The GRC problem introduces various spatial and temporal facets dealing with crossing gate behavior and motor vehicle traffic through the crossing. Although many rail crossing problems contain similar components [5, 6, 10], this paper uses the basic syntax described in [6], which is summarized below.

Figure 2 shows a GRC example, where I is the crossing, P is a portion of track before I and R is the region of interest for train r. Trains are denoted r, r', r'', ... as needed. To aid in the discussion of the shape calculus implementation, a discrete numerical representation is added, where the region I corresponds to the interval $[0, 2)$ and P corresponds to the interval $[2, 10)$. The GRC problem provides the opportunity to investigate simple safety and liveness/utility functions that ensure that the gate is down when a train is in the crossing and that the gate returns to the up position when no train is present. These properties are expressed through the specification languages used for model checking or runtime assertions as described in the following sections.

3. Sparse Space-Time

Specification languages come in many varieties (e.g., automata, state transition diagrams and logics) and can be used to model aspects of physical implementations. Often, the physical characteristics of an implementation can introduce additional system constraints that are not traditionally present within current reasoning. For example, vehicles are constrained to specific rates of change (e.g., acceleration/deceleration) and system state sampling dictates what can be observed and what might be missed. In order to incorporate physical property constraints within the specification language, we couple these aspects within the spatial-temporal *next* operator. This approach is inspired by Kopetz's work with dense and sparse time [8] and naturally fits with the system sampling frequency in that each *next* state corresponds to a state capture of the physical system. In Kopetz's original work, real time (dense time) is constrained to

sparse time intervals such that ordering properties can be maintained in a distributed system with respect to allowable global clock drift. In this approach, events occurring in real time (but after a sparse time interval) are recorded in the next sparse time interval. In our work, the true occurrence of events (in real time) can only be observed at discrete system state capture intervals (the equivalent of sparse time).

The goal of sparse space-time is twofold: (i) fit naturally with the sparse temporal frequency of system state collection; and (ii) provide the capability to reason in the sparse spatial domain (RCC) without risking the loss of dense state transitions. For example, when disconnect holds at one system state capture and the spatial relationship transitions to edge connect and then to partially overlapping before the next system state capture, it is necessary to account for the occurrence of both edge connect and partially overlapping relationships. Without a proper mechanism, sparse space-time in our approach, the system may not be able to recognize when a spatial relationship is satisfied.

This concept is not necessarily new. Gerevini and Nebel [3] refer to the general notion as the "continuity constraint." However, extending this baseline concept with the wider breadth of available actions that occur between state captures introduces an opportunity to include implementation constraints in the logic that can abstract away some of the complexity and allow for the more natural expression of assertions. In this way, physical properties can be included within baseline specification pieces, and then separated from the other dynamics of desired system operation. This intuitive mechanism abstracts the physical system constraints in the specification language while maintaining the specific restrictive details.

Definition. The *next* operator in sparse space-time is defined as:

$$\sigma_i \models \mathcal{P}, \;\; \sigma_{i+1} \models \bigcirc \mathcal{P} \tag{1}$$

where σ is the spatial-temporal domain $\mathcal{T} \times \mathcal{S}$ and i denotes the temporal discretization.

In the sparse space-time approach, space is discretized among the included dimensions and the resulting spatial regions can be reasoned about using the discretization. The spatial discretization aspect becomes important with respect to the relationship between the spatial regions of interest, but mostly in the operational semantics regarding spatial-temporal transitions. The complementary aspect of our sparse space-time approach is the temporal discretization. The temporal discretization itself is fairly straightforward and can be coupled to the sparse space concept of sparse time intervals corresponding to system state collections with a sampling speed ($SamplingRate$). The innovative aspect incorporates the physical properties (e.g., sampling rate) with the discretization knowledge through Equation (1) to provide additional system constraints that include limiting the available spatial-temporal transitions that occur during system state collection points.

Wolter and Zakharyaschev [12] note that the "next time" operator makes no sense in dense time flows such as $\{\mathbb{Q}, <\}$ or $\{\mathbb{R}, <\}$. This follows because any two real numbers will have some identifiable real number between them and, therefore, no quantifiable "next" state. The sparse space-time approach follows naturally and extends the notion of density and sparsity into the spatial realm. Shifting from a dense space to a sparse space equivalently shifts the spatial referencing into a uniform metric space, which remains a topological space. Thus, within sparse space-time, sparse space is handled as a metric space on some flow such as $\{\mathbb{N}, <\}$ or $\{\mathbb{Z}, <\}$. Remaining within a topological space, the fundamental spatial-temporal logic work of Wolter and Zakharyaschev [12] continues to hold, and each of the fundamental RCC-8 relationships can be represented in sparse space-time. However, it still remains to be shown whether or not a sparse space-time representation can represent the physical characteristics of a modeled system. In other words, is sparse space-time sound and complete with respect to expressing sparse truths of a dense physical system?

3.1 Sparse Space-Time Soundness

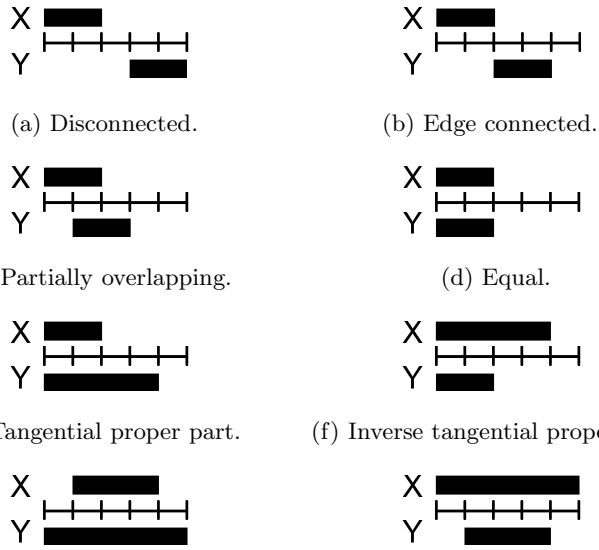
As with any new approach, it is important to understand the benefits and limitations. Logic systems often provide measures of the soundness and completeness as a fundamental baseline. Logical soundness expresses the property that the logic only proves formulas that are valid. The logical soundness of the spatial-temporal logic was addressed by Wolter and Zakharyaschev [12]. Since sparse space-time retains a spatial-temporal logic base and changes only the semantics of the *next* operator, the notion of sparse space-time soundness is taken to mean the ability to express every RCC-8 relationship as specified in Theorem 1 below.

Lemma 1 (Metric Space Uniqueness). *When implemented as a metric space with a simple temporal distance metric, each sparse space-time capture is unique.*
Proof: A simple metric space distance can be defined as:

$$d(p_1, p_2) = \sqrt{(t_2 - t_1)^2}$$

where distance d is calculated as the temporal difference between two points $p_1 = (x_1, y_1, z_1, t_1)$ and $p_2 = (x_2, y_2, z_2, t_2)$. Since the sparse space-time *next* operator is defined in Equation (1) with respect to the temporal discretization i, if $p_1 \neq p_2$, then $t_1 \neq t_2$. With all points unique, $t_1 \neq t_2$ holds for all pairwise comparisons, leading to a strict total ordering under $<$ of all sparse space-time events. □

Theorem 1 (Sparse Space-Time Soundness). *Every one-dimensional spatial-temporal relationship is expressible in one-dimensional sparse space-time.*
Proof: By definition, RCC-8 is jointly exhaustive and pairwise disjoint; this



(a) Disconnected. (b) Edge connected.

(c) Partially overlapping. (d) Equal.

(e) Tangential proper part. (f) Inverse tangential proper part.

(g) Nontangential proper part. (h) Inverse nontangential proper part.

Figure 3. Expressing spatial RCC-8 representations in sparse space-time.

means that all spatial situations can be expressed as one of these relationships. Lemma 1 provides the complementing temporal expression through a total temporal ordering of sparse space-time. This result, coupled with showing that sparse space-time can express all the RCC-8 relationships, demonstrate the soundness of the expression in sparse space-time. Figure 3 expresses all the RCC-8 relationships in sparse space-time; this demonstrates soundness. □

Interval temporal logic (ITL) is a temporal logic designed around intervals of time and relationships between the intervals [1]. ITL expresses qualitative relationships between two intervals of time through seven relations and thirteen interval relationships. The equality (*equal*) relationship is reflexive, which eliminates its dual, yielding thirteen relationships from seven temporal interval relations. Figure 3 shows how one-dimensional RCC-8 representations naturally map to ITL relations: X *before* Y (Figure 3(a)), X *meets* Y (Figure 3(b)), X *overlaps* Y (Figure 3(c)), X *equals* Y (Figure 3(d)), X *starts* Y (Figure 3(e)), Y *starts* X (Figure 3(f)), X *during* Y (Figure 3(g)), and Y *during* X (Figure 3(h)).

In general, multiple ITL relations can map to a single RCC-8 relationship, e.g., X *before* Y and Y *before* X both represent DC(X,Y). Reducing the set of ITL expressions to a subset that represents RCC relationships yields: BEFORE, MEETS, OVERLAPS, EQUAL, STARTS/FINISHES, STARTS^{-1}/ FINISHES^{-1}, DURING and DURING^{-1}. Syntactically, DURING indicates X

(a) Equal objects. (b) Nontangential proper part.

Figure 4. Dense EQ and NTPP mappings to sparse EQUALS.

during Y and DURING^{-1} indicates Y *during* X for DURING, STARTS and FINISHES in the discussion of dense and sparse mappings.

3.2 Sparse Space-Time Completeness

Logical completeness expresses the property that any stated proposition can be proven true or false. For sparse space-time to be complete, any statement expressible in sparse space-time would have to be provable in dense space. Stated another way, the completeness of sparse space-time refers to the ability to map between dense and sparse space. A simple example shows that a $1:1$ mapping is not possible.

Consider a one-dimensional sparse space representation where a sparse spatial discretization is flagged if any part of the dense spatial object occupies a portion of the sparse space. Two dense spatial conditions can result in the same sparse spatial representation as shown in Figure 4. Thus, the sparse spatial representation is insufficient to show a unique dense spatial configuration.

Table 2. Mappings from dense RCC relationships to intervals.

RCC Relationship	**Possible Interval Relationships**
DC	BEFORE, MEETS
EC	MEETS
PO	STARTS^{-1}/FINISHES^{-1}, OVERLAPS
EQ	EQUAL
TPP	STARTS/FINISHES, EQUAL
NTPP	STARTS/FINISHES, EQUAL, DURING
TPP^{-1}	STARTS^{-1}/FINISHES^{-1}
NTPP^{-1}	DURING^{-1}

It is assumed that a region of interest must be at least as large as a single spatial discretization. If this were not the case, any RCC-8 relationship could occur within a spatial discretization, allowing no knowledge of the dense spatial relationship. Working in the single spatial dimension, the sparse space representations map naturally to Allen's temporal interval relationships [1]. Interested readers are referred to [4] for a complete analysis of the mapping along with all the full proofs. Table 2 summarizes all the possible mappings between dense RCC-8 relationships and one-dimensional sparse space-time representations.

Theorem 2 (Sparse Space-Time Completeness). *All dense one-dimensional space-time can be represented in one-dimensional sparse space-time.*
Proof: The proof follows directly from Table 2. □

Lemma 2. *The one-dimensional sparse space-time relationships DURING, DURING^{-1}, BEFORE and OVERLAPS uniquely map back to dense space.*
Proof: The proof follows directly from Table 2. □

Theorem 3 (Transitions). *If it can be shown that the EQ RCC relationship will never hold (e.g., by assuring that spatial objects will never be the exact same size), then the remaining one-dimensional dense space transitions can be captured in sparse space-time.*

Proof: Eliminating the RCC-8 transitions through EQ yields three possible transition paths stemming from PO: (i) between PO and DC; (ii) between PO and NTPP; and (iii) between PO and NTPP^{-1}. From Lemma 2 and Table 2, OVERLAPS uniquely maps back to PO, meaning that OVERLAPS can uniquely identify a one-dimensional dense space PO relationship. Likewise, BEFORE maps to DC, DURING maps to NTPP and DURING^{-1} maps to NTPP^{-1}. Therefore, due to Theorem 2, transitions between any two sparse space-time observations of OVERLAPS, BEFORE, DURING and DURING^{-1} lead to a path that guarantees a dense RCC-8 relationship held between observations. □

From Theorem 3, four sparse space-time states can account for dense states and transitions of seven of the eight RCC-8 relations. While it is not possible to completely map from the sparse space-time domain into dense space, these properties allow for RCC reasoning within one-dimensional sparse space-time.

4. Sparse Space-Time GRC Specification

Having defined sparse space-time, we can now attempt to specify the generalized railroad crossing within the framework. The specification begins by capturing the dynamics of gate operation, because the position of the gate determines if the overall system is safe.

Gate operation can be specified in a number of ways. We treat the gate almost as a separate entity because it has a spatial relationship between its opening and closed positions, which could be thought of as separate from the one-dimension of the rail crossing. Using the notation that *gate* indicates the gate position in radians, where $gate = 0$ indicates that the gate is down and $gate = \frac{\pi}{2}$ indicates the gate is up, a minimum gate speed GS_{min} and maximum gate speed GS_{max} in rev/s can be incorporated to express gate opening and closing. These measurements could be transformed back into the straight line one-dimensional characteristics. However, the circular nature of gate operation

fits more naturally with circular parameters that are perfectly acceptable in the sparse space-time approach.

Given the proper part (PP) specified as:

$$PP(\alpha, \beta) = TPP(\alpha, \beta) \vee NTPP(\alpha, \beta).$$

and constraining the gate to only operate between $gate = 0$ (satisfying the Boolean DOWN) and $gate = \frac{\pi}{2}$ (satisfying UP), the minimum and maximum distances of gate travel are defined using the spatial region extending from:

$$\gamma = gate + \frac{GS_{min} \times 2\pi}{SamplingRate} \text{ to } gate + \frac{GS_{max} \times 2\pi}{SamplingRate}.$$

The following specification captures the opening gate operations, beginning with the initial transition:

$$DOWN \wedge \bigcirc \neg DOWN \Rightarrow PP(\bigcirc gate, \gamma) \wedge GoingUp.$$

This equation states that if the gate is changing from the down position, then it must travel at least the distance traveled at the minimum gate speed during the system state capture and no more than the maximum distance.

Additional variables are introduced to ensure that the gate condition is "going up" or "going down" to ensure continuation of motion. The specification to capture the gate movement from DOWN to UP is quite similar:

$$GoingUp \wedge \neg UP \Rightarrow \bigcirc(UP \wedge \neg GoingUp) \vee PP(\bigcirc gate, \gamma).$$

Expressing the gate closing operation follows similarly using:

$$\delta = gate - \frac{GS_{min} \times 2\pi}{SamplingRate} \text{ to } gate - \frac{GS_{max} \times 2\pi}{SamplingRate}$$

as the spatial region that the gate will maintain during the following state capture given the speed bounds.

The initial transition is captured as:

$$UP \wedge \bigcirc \neg UP \Rightarrow PP(\bigcirc gate, \delta) \wedge GoingDown$$

with the subsequent motion confined by:

$$GoingDown \wedge \neg DOWN \Rightarrow \bigcirc(DOWN \wedge \neg GoingDown) \vee PP(\bigcirc gate, \delta).$$

A final safety bound constrains the gate to be between the UP and DOWN positions as constant $PP(gate, [DOWN, UP])$, meaning that the gate must always remain somewhere between (but inclusive of) the range $[0, \frac{\pi}{2}]$. This somewhat tedious specification of gate operation now makes it possible to capture much more powerful aspects of the GRC safety and liveness than with more concise equations.

4.1 GRC Safety

As shown in Figure 2, r, P, I and R represent the spatial entities train, pre-crossing, in-crossing and region of interest (where $R = P + I$). The initial safety specification is expressed in spatial-temporal logic as:

$$\neg DC(r, I) \Rightarrow DOWN$$

which requires that the gate be down whenever a train is present in the crossing area I. To obtain this safety property, the entry of the train into region P can trigger the lowering of the gate as in:

$$DC(r, P) \wedge \bigcirc \neg DC(r, P) \Rightarrow GoingDown.$$

This assertion in combination with:

$$GoingDown \wedge \neg DOWN \Rightarrow \bigcirc(DOWN \wedge \neg GoingDown) \vee PP(\bigcirc gate, \delta)$$

preserves GRC safety in sparse space-time.

All the typical non-Byzantine assumptions are carried over as well – trains are not spaced so close that the gate has to be raised and lowered before a car can proceed through the crossing, cars obey the gate signals, etc. Furthermore, the distance P could be checked against the gate and train speeds to ensure appropriate safety in that the gate must go from UP to DOWN in no more time than it takes for a train to travel distance P. The minimum and maximum values of train speed are denoted by TS_{min} and TS_{max}, respectively. To verify that the gate has enough time to be lowered between the detection of entry into P, the shortest time for the train to traverse the distance P must be greater than the longest time it takes the gate to be completely raised. In other words:

$$\frac{P}{TS_{max}} > \frac{1}{4 \times GS_{min}}$$

where TS_{max} is in m/s, P is in m, and GS_{min} is in rev/s.

4.2 GRC Liveness

The liveness restriction prevents the gate from being down when no trains are present. This is expressed in spatial-temporal logic as:

$$DC(r, R) \Rightarrow \neg DOWN.$$

Satisfying liveness requires the gate and train dynamics as presented above in the discussion of safety:

$$GoingUp \wedge \neg UP \Rightarrow \bigcirc(UP \wedge \neg GoingUp) \vee PP(\bigcirc gate, \gamma)$$

with the initiating condition:

$$\neg DC(r,R) \wedge \bigcirc DC(r,R) \Rightarrow \bigcirc GoingUp.$$

These fulfill the liveness property with the condition that one additional system state capture may be required when disconnect is detected but the gate has not begun ascending. In other words, detection (or prediction) of $DC(r,R)$ is a prerequisite to beginning the $GoingUp$ sequence of events. Thus, at the system state capture where $DC(r,R)$ is first satisfied, $DOWN$ may also hold. However, at the next system state capture (per $GoingUp$), $\neg DOWN$ will hold. This nuance in the system dynamics is one area where specifications can become bogged down in the details of the language instead of the "big picture" aspects of the specification. However, the combination of such nuances (e.g., sampling frequency, sampling errors and measurement variations) combine to produce formidable challenges to understanding the limits and capabilities of real-world implementations of runtime assertion checking.

5. Comparison of Specifications

Given the GRC example described above, the correct system operation can be discussed in two main ways: (i) an axiomatic specification that prescribes the allowed behavior; and (ii) an operational specification that describes system operation conforming to the allowable behavior. The axiomatic specification comprises invariants that hold over correct system performance. The operational specification provides the mechanistic requirements that produce the desired behavior. Our analysis focuses on the operational specifications of generalized safety and liveness properties, including factors that may impact the human understanding of assertions in each specification instance. The specifications are compared based on the number of required initial and supporting definitions, number of equations and depth of expressions enumerated as a single count of tokens (e.g., comparisons and calculations). These metrics were selected to represent the amount of estimated effort to understand and generate subsequent assertions within the languages.

5.1 Original GRC Specification

The original GRC paper by Heitmeyer and Lynch [6] examined variations of a railroad crossing example and rebuilt the problem from the ground up. Their approach focuses on an axiomatic specification of the safety and liveness properties along with operational specifications of the trains and gates, incorporating timed automata, invariants and simulation mappings to model and verify correct system behavior. Reproducing even a portion of their work here for illustrative purposes is unreasonable due to the iterative nature of their formal methods and the amount of material that is required to show correctness. However, a hint of their process is included to illustrate the comparative metrics.

Table 3. Automaton excerpt.

State	**Transitions**
now	$enterR(r)$
for each train r:	Precondition:
$r.status$	$s.r.status = not\_here$
$first(enterI(r))$	Effect:
$last(enterI(r))$	$s'.r.status = P$
	$s'.first(enterI(r)) = now + \epsilon_1$
	$s'.last(enterI(r)) = now + \epsilon_2$

The Heitmeyer-Lynch axiomatic specification, represented as timed automata, is an initial step in formal verification and helps provide comparative metrics. In the specification, system variables include the upper and lower timing bounds (e.g., time to raise the gate); definitions include listed restrictions such as a lower bound system variable is less than or equal to the corresponding upper bound; and equations, which are transition preconditions or effects in the automaton and safety and liveness specifications. Tokens do not have an exact counterpart in the other GRC specifications, but are obtained from the states and transitions where each comparison or calculation is a token.

As an example, consider the automaton excerpt in Table 3. The excerpt has the system variables ϵ_1 and ϵ_2 (defined previously), four equations (all under transitions) and fourteen tokens. Note that each state, equation operation and reference is considered to be a token. Thus, now and $r.status$ are both calculated as a single token whereas $first(enterI(r))$ and $last(enterI(r))$ are both counted as two tokens because they reference $enterI(r)$.

Table 4. Heitmeyer-Lynch GRC specification.

System Variables	8
Definitions	4
Equations	30
Tokens	89

For the purpose of counting tokens, it is assumed that there is only one train r under the "for each train" in state. In the case of the token counts corresponding to the remaining transitions, the precondition is a single token, the first effect is a single token and the final two effects are each three tokens (containing two equation operations and one reference). Performing these computations over their entire specification yields the results shown in Table 4.

5.2 Shape Calculus GRC Specification

The second model used for comparison is Quesel and Schafer's shape calculus [10]. This specification incorporates discrete time and space, allowing for finite space and infinite time (these choices are due to decidability issues that allow model checking). The discretization corresponds to the numeric discretization of the railroad shown in Figure 2. Key aspects of the shape calculus are included here to assist in understanding the notation and how comparative metrics relate to the GRC specifications.

An observation of a train at a specific point in time and space is *train*; if this holds at all points in time and space (in the interval of interest), then it is expressed as $\lceil \text{train} \rceil$. The "chop" operator, where $\mathcal{F}\langle e_d \rangle \mathcal{G}$ reads $\mathcal{F}$ chop $\mathcal{G}$, specifies that there is a chop point in dimension d (time and/or space) at which $\mathcal{F}$ holds up to and including the chop point, and $\mathcal{G}$ holds at and after the chop point. The diameter of the spatial or temporal dimension d is ℓ_{e_d} and an empty observation interval $\lceil \rceil_{e_d}$ has a diameter of zero. The "somewhere" operator $\Diamond_{e_d}\mathcal{F}$ is defined if and only if $\mathcal{F}$ is true in some subinterval:

$$\Diamond_{e_d}\mathcal{F} \equiv true\langle e_d \rangle \mathcal{F} \langle e_d \rangle true.$$

The globally operator must hold in all subintervals, expressed as the dual:

$$\Box_{e_d}\mathcal{F} \equiv \neg\Diamond_{e_d}\neg\mathcal{F}.$$

This basic syntax can be used to express the shape calculus propositions corresponding to the GRC.

The most basic proposition determines if there is a train within an interval. Using the definitions in [10], this is defined as trainPartWeak:

$$\text{trainPartWeak} \equiv \neg(\lceil \neg\text{train} \rceil \langle e_x \rangle true).$$

However, this must be strengthened to exclude the empty observation interval, so:

$$\text{trainPart} \equiv \text{trainPartWeak} \wedge \ \ell_{e_x} > 0.$$

Or,

$$\text{trainPart} = \neg(\lceil \neg\text{train} \rceil \langle e_x \rangle true) \wedge \ \ell_{e_x} > 0.$$

A distance operator dist(δ) is defined in [10] as:

$$\text{dist}(\delta) \equiv ((\lceil \neg\text{train} \rceil \vee \lceil \rceil_{e_x}) \wedge \ell_{e_x} = \delta) \ \langle e_x \rangle \text{trainPart}.$$

This divides the track so that the rightmost part contains no train and the leftmost part contains the train, providing a measure of distance δ from the train to the end of the observation interval. Expanding this based on first principles yields:

$$\text{dist}(\delta) \equiv ((\lceil \neg\text{train} \rceil \vee \lceil \rceil_{e_x}) \wedge \ell_{e_x} = \delta)\langle e_x \rangle (\neg(\lceil \neg\text{train} \rceil \langle e_x \rangle true) \wedge \ \ell_{e_x} > 0).$$

A careful observation of the propositions shows that $\mathcal{F}$ holds $\lceil\neg$ train$\rceil$ and $\mathcal{G}$ holds $\neg\lceil\neg$ train$\rceil$ (or $\neg\mathcal{F}$). However, at the chop point, both $\mathcal{F}$ and $\mathcal{G}$ must hold, which requires $\mathcal{F} = \neg\mathcal{F}$. Ignoring this technicality and assuming that the shape calculus distance operator holds, the regions of the track can be specified as empty $\equiv \lceil\neg$ train$\rceil$, appr $\equiv dist(\delta) \wedge 10 > \delta \geq 2$, and cross $\equiv dist\delta \wedge 2 > \delta$. Thus, relating these back to the regions in Figure 2, "empty" indicates that no train is present in the region of interest R, "appr" indicates that there is a train in P (approaching) and "cross" indicates that a train is in the crossing region I.

To maintain the physical properties of the system such as restricting the train speed to under the maximum limit and preventing the train from stopping in the crossing indefinitely, Quesel and Schafer introduce a runProgress specification defined as:

$$\text{runProgress} \equiv \Box_{e_t}\Box_{e_x}(((l_{e_x} = MAXSPEED\langle e_x\rangle\text{trainPart}) \wedge l_{e_t} = 1) \langle e_t\rangle l_{e_t} = 1) \Rightarrow (l_{e_t} = 1\langle e_t\rangle\text{trainPart}).$$

Expanding according to the first principles, yields:

$$\begin{aligned}\text{runProgress} \equiv \neg(true\langle e_t\rangle(true\langle e_x\rangle\neg(((l_{e_x} = MAXSPEED\langle e_x\rangle \\ \neg(\lceil\neg\text{train}\rceil\langle e_x\rangle true) \wedge \ell_{e_x} > 0) \wedge l_{e_t} = 1)\langle e_t\rangle l_{e_t} = 1)\langle e_x\rangle true)\langle e_t\rangle true) \\ \Rightarrow (l_{e_t} = 1\langle e_t\rangle\neg(\lceil\neg\text{train}\rceil\langle e_x\rangle true) \wedge\ \ell_{e_x} > 0).\end{aligned}$$

Additional specifications are necessary to complete the shape calculus specification of safety and liveness. However, these initial specifications suffice to illustrate the format of GRC specifications and how they are incorporated within the comparisons.

The definitions in the specification include the chop operator $\mathcal{F}\langle e_d\rangle\mathcal{G}$, diameter $\lceil\ \rceil_{e_d}$ and negation $\neg\mathcal{F}$. Each usage constitutes a token along with standard binary operators ($\wedge$, $\vee$, $>$, $<$, $=$, $\Rightarrow$). Thus, the Equation trainPartWeak given by:

$$\neg(\lceil\neg\text{train}\rceil\langle e_x\rangle true)$$

contains four tokens: the encompassing negation, chop operator, diameter and final negation. Note that train and true are base expressions and not evaluated as tokens. Any equation incorporating other expressions automatically brings in the accompanying tokens (as would be the case when expanded according to first principles). Thus, the specification of trainPart:

$$\text{trainPart} \equiv \text{trainPartWeak} \wedge\ \ell_{e_x} > 0$$

contains six tokens: four from trainPartWeak, and one each from the *and* operator and greater-than comparison. It is interesting to note that, in this specification, the two system variables are MaxSpeed and ReactTime, which are both translated back to the GRC example to relate them to the quantitative,

Table 5. Quesel-Schafer GRC specification.

System Variables	2
Definitions	5
Equations	11
Tokens	129

physical spatial distance that the train can travel during a temporal interval. Table 5 shows the results for the Quesel-Schafer GRC specification.

5.3 Sparse Space-Time GRC Specification

Summarizing the discussion above, the sparse space-time GRC specification holds when the following assertions are satisfied:

$$PP(gate, [DOWN, UP]) \tag{2}$$

$$DC(r, P) \wedge \bigcirc \neg DC(r, P) \Rightarrow \bigcirc GoingDown \tag{3}$$

$$GoingDown \wedge \neg DOWN \Rightarrow \bigcirc (DOWN \wedge \neg GoingDown) \vee PP(\bigcirc gate, \delta) \tag{4}$$

$$\neg DC(r, R) \wedge \bigcirc DC(r, R) \Rightarrow \bigcirc GoingUp \tag{5}$$

$$GoingUp \wedge \neg UP \Rightarrow \bigcirc (UP \wedge \neg GoingUp) \vee PP(\bigcirc gate, \gamma) \tag{6}$$

Equation (2) is a safety constraint that binds the gate between the down and up positions. Equations (3) and (4) initiate and facilitate the lowering of the gate upon train entry. Equations (5) and (6) initiate and facilitate the raising of the gate upon train exit. The sparse space-time system variables include *gate* and r, which denote the position of the gate and train, respectively, along with GS_{min}, GS_{max}, *SamplingRate*, and the Boolean variables *GoingUp* and *GoingDown*. The definitions include the spatial regions P, R, δ and γ. Finally, each RCC relation or operator is counted as a token (e.g., $PP(\bigcirc gate, \gamma)$ is two tokens, one for the Proper Part relation and one for the *next* operator).

5.4 Discussion

Table 6 presents a comparison of all the GRC specifications. The comparison indicates that the sparse space-time approach is more terse than the other specifications based on the system variables, definitions and equations, and counting the number of tokens required to express safety and liveness.

Assertion checking is based on a logical specification of correct operating behavior. It is anticipated that properties of critical infrastructure systems can be incorporated within assertion checking as additional, inherent constraints, which is how expressions can capture system dynamics natively within sparse

Table 6. Comparison of GRC specifications.

GRC Specification	Vars.	Defns.	Eqns.	Tokens
Heitmeyer-Lynch	8	4	30	89
Quesel-Schafer	2	5	11	129
Sparse Space-Time	7	4	5	32

space-time. The result is that assertions can be much more honed to the desired properties (e.g., safety and liveness) and do not have to be explicitly crafted to capture or convert physical properties such as the speeds of gates and trains in the GRC example.

6. Conclusions

The sparse space-time paradigm can be used to naturally express spatial-temporal assertions pertaining to critical infrastructure systems. The utility of the paradigm is demonstrated via a specification of a one-dimensional railway crossing problem, which includes safety and liveness properties. The generalized metrics indicate that, although they may be more terse, sparse space-time assertions are actually simpler to understand and create.

Our future research will explore the application of the sparse space-time paradigm in domains that span multiple dimensions and/or include the modeling of cyber-physical systems. Additionally, we will consider space-time trajectories as fundamental components of the spatial-temporal *next* operator; this will help reduce the dependence on external motion specifications, such as those used for gate dynamics. Our future research will also examine the application of the flow tree concept [13] in conjunction with the sparse space-time paradigm.

Acknowledgements

This research was supported in part by the Future Renewable Electric Energy Distribution Management Center, an NSF Engineering Research Center, under Grant No. EEC 0812121; and by the Missouri S&T Intelligent Systems Center.

References

[1] J. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM*, vol. 26(11), pp. 832–843, 1983.

[2] B. Clarke, A calculus of individuals based on "connection," *Notre Dame Journal of Formal Logic*, vol. 22(3), pp. 204–218, 1981.

[3] A. Gerevini and B. Nebel, Qualitative spatio-temporal reasoning with RCC-8 and Allen's interval calculus: Computational complexity, *Proceedings of the Fifteenth European Conference on Artificial Intelligence*, pp. 312–316, 2002.

[4] M. Gosnell and B. McMillin, Sparse Space-time Specification of the Generalized Railroad Crossing, Technical Report CSC-12-01, Department of Computer Science, Missouri University of Science and Technology, Rolla, Missouri, 2012.

[5] A. Haxthausen and J. Peleska, Formal development and verification of a distributed railway control system, *IEEE Transactions on Software Engineering*, vol. 26(8), pp. 687–701, 2000.

[6] C. Heitmeyer and N. Lynch, The generalized railroad crossing: A case study in formal verification of real-time systems, *Proceedings of the Real-Time Systems Symposium*, pp. 120–131, 1994.

[7] R. Kontchakov, A. Kurucz, F. Wolter and M. Zakharyaschev, Spatial logic + temporal logic = ? in *Handbook of Spatial Logics*, M. Aiello, I. Pratt-Hartmann and J. van Benthem (Eds.), Springer, Dordrecht, The Netherlands, pp. 497–564, 2007.

[8] H. Kopetz, Sparse time versus dense time in distributed real-time systems, *Proceedings of the Twelfth International Conference on Distributed Computing Systems*, pp. 460–467, 1992.

[9] N. Malik, J. Baumgartner, S. Roberts and R. Dobson, A toolset for assisted formal verification, *Proceedings of the IEEE International Conference on Performance, Computing and Communications*, pp. 489–492, 1999.

[10] J. Quesel and A. Schafer, Spatio-temporal model checking for mobile real-time systems, *Proceedings of the Third International Colloquium on the Theoretical Aspects of Computing*, pp. 347–361, 2006.

[11] D. Randell, Z. Cui and A. Cohn, A spatial logic based on regions and connection, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pp. 165–176, 1992.

[12] F. Wolter and M. Zakharyaschev, Qualitative spatiotemporal representation and reasoning: A computational perspective, in *Exploring Artificial Intelligence in the New Millennium*, G. Lakemeyer and B. Nebel (Eds.), Morgan Kaufmann, San Francisco, California, pp. 175–216, 2002.

[13] J. Wood, J. Dykes, A. Slingsby and R. Radburn, Flow trees for exploring spatial trajectories, *Proceedings of the Seventeenth Annual Conference on GIS Research*, pp. 229–234, 2009.