Spatiotemporal Trajectory Models for Metalevel Target Tracking

Mustafa Fanaswala, Vikram Krishnamurthy
University of British Columbia
Vancouver, BC, Canada

INTRODUCTION

Physical sensor–based target tracking is a classical problem that has been studied in great detail [1], [2]. This article presents metalevel tracking middleware algorithms to help human radar operators interpret tracks in order to detect and visualize suspicious spatiotemporal target trajectories. While state space models are ideal for target tracking, the main idea in this article is that stochastic context-free grammar (SCFG) models are also useful for modeling and interpreting trajectories.

In our previous articles [3], [4], several SCFG models were presented for specific target trajectories signifying malicious intent. The current article generalizes these trajectory models and extends them toward novel multitarget anomalous patterns. More specifically, eight trajectory models are presented: (a) random walk, (b) reciprocal process, (c) linear, (d) arclike, (e) rectangular, (f) destination-aware, (g) palindromic, and (h) target rendezvous trajectories. Our modeling framework focuses on the human–sensor interface (middleware) tasked with high-level reasoning and visualization from lower-level sensor measurements. Bayesian signal processing algorithms are also developed to perform model classification and change detection using novel SCFG models.

EXAMPLE

Consider the scenario in Figure 1, where a target executing a trapezoidal trajectory (dashed blue line) is observed in noise. This trajectory could signify a target avoiding an obstacle by deviating away, passing the obstacle, and then returning to its previous path. Given noisy point measurements, how can one devise algorithms to detect whether the target executed such a pattern? This is a nontrivial estimation problem (it cannot be solved efficiently with template matching techniques), because exponentially many–scaled versions of the shape need to be considered.1 SCFG models offer an efficient framework to model common shapes (Figure 2a) in a scale-invariant manner.

MOTIVATION: METALEVEL TRACKING AND INTENT

Inference

Classical target tracking assumes a Markovian state space model for the target kinematics. Such models are useful on short timescales (on the order of several seconds), and many well-known target tracking algorithms based on such Markov models have been developed in the literature [1]. This article is motivated by metalevel target tracking applications on longer timescales (on the order of several minutes). In metalevel tracking, one is interested in devising automated procedures that assist a human analyst to interpret the tracks obtained from a conventional tracking algorithm. On such longer timescales, most real-world targets are driven by a premeditated intent. The intent of a target can manifest in the shape of the target trajectory or its final destination (among other attributes). In this article, trajectory shape is modeled using SCFG models that are ideally suited to model patterns due to scale invariance, which emerges from their self-embedding properties. Moreover, time-varying SCFGs are used in this article to reflect the intent of the target

1 As an equivalent example, consider a string al bm cm dn, where l, m, and n are unknown positive integers such that l + 2m + n = k. How can one extract the arc trajectory bm cm from a noisy version of the string? If b and c were composite alphabets that were the union of r other symbols, template matching would require listing an exponential number \( \sum_{r=1}^{N} [k - (2m - 1)] r^{N-1} \) of possibilities for all possible choices of l, m, and n, since the values of these integers are not known.
to move toward its destination. Such a characterization emerges from the observation that local target dynamics at the metalevel may be anticipative (destination aware) or noncausal to terminate a trajectory in a known destination. Our previous article [3] utilized reciprocal stochastic models for such destination-aware trajectories, which are generalized in this article using time-varying SCFGs. The use of SCFGs in metalevel tasks is mainly motivated by their ability to capture arbitrary-range (as opposed to fixed-length) long-term dependencies in the target trajectory [5]. Consequently, metalevel analysis of target trajectories can serve as a visualization tool for suspicious trajectories and anomalous target behavior.

LITERATURE SURVEY

The classification and tracking of anomalous spatiotemporal trajectories arise in many application areas, such as target tracking using radars [3], gesture recognition using optical [6] and time-of-flight [7] sensors, human action recognition in camera networks [8], gait analysis [9], network packet traces [10], and vehicular geoposition coordinates [11]. The maritime surveillance literature has also recently seen an interest in trajectory anomaly detection applications [12], [13]. In particular, [13] examined the target rendezvous problem. A common approach in such sequential pattern recognition problems is the use of HMMs to capture the temporal dependence of observations. Such an approach was taken in [14] by using flow vectors on objects being tracked in video sequences. Spatial nodes of interest are first isolated using a Gaussian mixture modeling technique. Routes are then created by clustering different trajectories, and a high-level HMM is learned for each route. However, such a model cannot incorporate destination-specific information. Moreover, long-term dependencies in the trajectory are lost due to the Markov assumption.
Our article is related to the approach taken in [15] and [16]. A nonprobabilistic context-free grammar approach was used in [15] to identify two-person interactions like hugs, handshakes, kicks, and punches to enforce syntactic structure on detected events. In [16], SCFGs were used to recognize cheating actions in card games at casinos. Our article departs from them significantly as we consider trajectory modeling in a tracking situation and not an action recognition system. This article generalizes the treatment in [3] and presents novel models for multitarget anomalous trajectories.

Multitarget tracking (MTT) has a rich literature comprising various powerful tracking algorithms like the joint probabilistic data association [17], interacting multiple-model filter [1], multiple hypothesis tracking [18], and probability hypothesis density filter [19]. A taxonomy of MTT approaches is provided in [20]. The proposed work is not an alternative to a comprehensive tracker but a complementary tool to pick out anomalous trajectories at a coarser scale. Tracking errors manifesting as missed or spurious detections, incomplete or false tracks, and errors in data association are assumed to have originated and/or missed or spurious detections, incomplete or false tracks, and errors in data association are assumed to have originated and/or been corrected at the fine timescale. Such errors are not explicitly treated in this article but are regarded as noise parameters, which can be incorporated into the higher-level logic. Previous articles [21] and [22] dealt with some of these tracker limitations more explicitly.

**Hierarchical Tracking Framework to Assist Human Operators**

In this section, a system-level description of the tracking framework is presented (Figure 3). A conventional (base-level) target tracker operates on the fast timescale, while the higher-level middleware layer operates on a slower timescale.

**Base-Level Tracker**

The base-level tracker is a Bayesian filter operating on the fast timescale (order of seconds) denoted by \( t = 1, 2, \ldots \). The base-level tracker can be represented as an operator \( T \) that uses sensor measurements \( z_t \) to update a posterior filtering distribution \( F_t \) over the position and velocity of the target by

\[
F_t = T(F_{t-1}, z_t).
\]

For example, consider a target represented by its kinematic state \( s = [s^x, s^y, v^x, v^y]^T \), the state variables \( (s^x, s^y) \) refer to the position of the target, while \( (v^x, v^y) \) refer to the velocity of the target in Cartesian coordinates. Classical target tracking uses a state space model

\[
s_{t+1} = f(s_t, w_t), \quad z_t = h(s_t) + v_t,
\]

where \( w_t \) and \( v_t \) represent the state and the measurement noise, respectively. The state transition and measurement functions are represented by \( f(\cdot) \) and \( h(\cdot) \), respectively. A base-level tracker estimates the target state trajectory from the radar measurements in a causal manner. This is a filtering problem involving computation of \( (f(\cdot), h(\cdot)) \) to the process and measurement distributions \( F_t = \{F_{t|0}, \ldots, F_{t|t-1}\} \), define a tracklet on the slow timescale as

\[
\hat{a}_t = \mathcal{H}(F_t) = \mathcal{P}\{a_t\}.
\]

Here, the tracklet \( \hat{a}_t \) denotes a quantization of an average state vector \( \bar{s}_t \) of the target obtained from the tracklet estimator at epoch \( t \) and can be viewed as a noisy version of the “true” underlying quantized position and/or unit velocity vector of the target denoted as \( a_t \).

Two types of tracklets are considered: (a) the position tracklets \( \hat{a}_{\text{pos}} \), which are output by the position tracklet estimator \( \mathcal{H} \),

**Figure 3.**

The proposed hierarchical tracking framework to assist human radar operators. A base-level tracker \( T \) outputs filtered state estimates at a fast timescale \( t \). The tracklet estimator aggregates state estimates from the base tracker and emits quantized tracklets \( \hat{a}_t \) at a slower timescale \( T \). A set of SCFG models for different threat scenarios is considered in the classification problem. The sequence of noisy tracklets \( \hat{a}_{ij} \) is fed into the Earley-Stolcke parser either to perform model classification to generate alarms or, as a visualization tool, to recover the suspicious trajectory.
and (b) the velocity tracklets \( \mathbf{v}_t \), which are quantized through \( \mathcal{H}_v \). Tracklets are used as syntactic subunits of target trajectories. The SCFG shape models utilize velocity tracklets as subunits of the trajectory shape, while SCFG destination-specific pattern models utilize position tracklets as subunits of a goal-directed trajectory (following a specific pattern of visited sites).

The position tracklet estimator operates on a discretized surveillance space \( \Lambda \) over which the target is observed. At each time instant, the position tracklet estimator quantizes the average \( (\tilde{x}_t, \tilde{y}_t) \) state estimates to the closest element \( (x_i, y_i) \in \Lambda \) on the discretized two-dimensional grid \( \Lambda \), as shown in Figure 4a. The velocity tracklet estimator utilizes the average velocity estimate \( (\tilde{v}_{\mathbf{x}}, \tilde{v}_{\mathbf{y}}) \) to find the direction of the target. The possible directions of motion of the target are quantized into eight radial cardinal directions from the set \( \mathcal{V} = \{ \bar{a} = -\pi, \bar{b} = -\frac{3\pi}{4}, \cdots, \bar{e} = 0, \bar{f} = \frac{\pi}{4}, \bar{g} = \frac{3\pi}{4} \} \).

The cardinal directions represented by mode are shown in Figure 4b. Each mode is represented with a lowercase letter under an arrow to denote that it is a unit directional vector.

The framework presented in this section and the prior section on the base-level tracker does not consider the effects of missed and/or spurious detections. The SCFG framework, however, can also be used to deal with such scenarios, as shown in [22]. A simpler approach based on modifying the grammar was presented in [21].

---

**TRAJECTORY CLASSIFICATION OBJECTIVE**

A target trajectory is associated with a specific spatiotemporal pattern depending on its shape and/or the pattern of sites visited by the target. Each trajectory is assumed to be generated by a model \( G_i \in \mathcal{G}, k = 1, \ldots, K \), where there are \( K = |\mathcal{G}| \) different types of anomalous patterns under consideration. Development of these models is the main idea of this article and is described in the next section. As a target moves in a region of interest, it generates tracklets \( \mathbf{a}_t, t = 0, 1, \ldots \). The anomalous trajectory classification task is then defined as finding the model \( G^* \in \mathcal{G} \) that has the highest probability of explaining the observed tracklet sequence \( \mathbf{a}_0, \ldots, \mathbf{a}_T \):

\[
G^* = \arg \max_{G \in \mathcal{G}} \mathbf{P}(G^* \mid \mathbf{a}_0, \ldots, \mathbf{a}_T).
\]

**SCFG MODELS FOR ANOMALOUS PATTERNS**

In this section, eight types of models for anomalous target trajectories are presented. Readers who are unfamiliar with the SCFG formalism should read the tutorial material in Appendix A. The motivation behind using SCFG models like arcs, rectangles, and palindromes is that such trajectories cannot be exclusively generated by Markov models. A formal proof of this assertion can be constructed using the pumping lemma for regular languages (HMMs) [23, 24].

**RANDOM WALK MODELS**

Random walks are the most elementary of Markov models for target trajectories evolving on a road network represented as a finite set of nodes \( \mathcal{V} = \{ v_1, v_2, \ldots, v_m \} \) (Figure 4a). The target position \( v_t \) can be represented using the bin \( v_t \in \mathcal{V} \) in which it resides at instant \( t \). A Markov chain model \( \mathcal{G}_{MC} \) of target dynamics over such a road network can be parameterized using a prior distribution \( \mathbf{P}(v_0) \) over the initial state \( v_0 \) at time \( t = 0 \) and a possibly time-varying transition matrix \( A(i, j) = \mathbf{P}(v_t = v_j \mid v_{t-1} = v_i) \). A transition matrix for such random walk models can be calculated based on physical distance between nodes. The state transition diagram of such a model is shown in Figure 5.

**RECIPROCAL MODELS FOR DESTINATION-AWARE TRAJECTORIES**

A human target following anomalous behavior rarely moves according to a random walk model. The sequence of actions taken by an intelligent agent is often premeditated on a global scale with random variation at the local scale. With this assumption in mind, the local dynamics of a target are more appropriately modeled using a reciprocal process; see also our previous article [3].

---

2 This implies that a Markov chain can generate a sample path that is an instance of a shape with some finite probability. However, it cannot do so with a probability of 1 unless the state space is extended artificially to the length of the trajectory.
The state transition diagram for target trajectories on a road network.

Figure 5.

A discrete-time reciprocal process \( a_t \in \mathcal{V}^m \) is a one-dimensional Markov random field with the noncausal property

\[
P = \{ a_t = v_j | a_{t-1} = v_i, a_{t-2} = v_k \},
\]

parameterized by the homogeneous three-point transitions \( Q(i, j, l) = P \{ a_t = v_j | a_{t-1} = v_i, a_{t-2} = v_k \} \) with \( v_i, v_j, v_k \in \mathcal{V}^m \). In urban environments, traffic information can be used to estimate the three-point transitions \( Q(i, j, l) \) between intersections in a road network. This is described in the section on single-target scenarios.

Using reciprocal dynamics, a destination-aware trajectory is defined by a priori fixing the final destination of the target \( a_t = v_j \). The resulting Markov bridge is characterized by a probability transition law

\[
P(a_t | a_{t-1}, a_{t-2}) = \frac{P \{ a_t = v_j | a_{t-1} = v_i, a_{t-2} = v_k \} \cdot P \{ a_t = a_{t-1} | a_{t-2} = v_k \}}{P \{ a_t = v_j | a_{t-1} = v_i \}}.
\]

which induces a backward recursion for time-varying two-point Markov transitions

\[
B^i_j(i, j) = \frac{Q(i, j, l)}{B^{i-1}_{j-1}(j, l)} \left( \sum_{l' \in \mathcal{V}^m} B^i_j(l', l) \right)^{i-1},\]

The second term on the right-hand side of (5) is the normalization constant. The time-varying transitions \( B^i_j(i, j) = P \{ a_t = v_j | a_{t-1} = v_i, a_{t-2} = v_k \} \) refer to two-point transitions of a target with final destination \( a_T = v_j \). To ensure that the final destination of the target \( a_T = v_j \), we initialize the recursion with \( B^{i,k}_{j,k}(i, j) = 1.0 \) for \( j = k \) and 0.0 otherwise. In addition, at time \( T = 2 \), the target transitions according to \( B^{i,k}_{j,k}(i, j) = Q(i, j, k) \). A more detailed treatment can be found in [3].

### Destination-Aware Paths

A destination-aware path is a target trajectory that is heading toward a known destination \( a_T = v_j \) while following local dynamics according to the three-point transitions \( Q(i, j, l) \). The Markov bridge approach can also be viewed as a time-varying SCFG for destination-specific trajectories.

A destination-aware trajectory satisfies the constraint \( P \{ a_t = v_j \} = 1.0 \). A destination-aware SCFG model can be defined using a starting rule of the form \( S \rightarrow v_j \cdot P \) with rule probability \( P \{ S \rightarrow v_j \cdot P \} = P \{ a_t = v_j | a_T = v_j \} \). The destination-constrained SCFG model is characterized by rules of the form \( X_{i,j} \rightarrow v_j X \) with time-varying rule probabilities given by the Markov bridge transitions such that \( P \{ X_{i,j} \rightarrow v_j X \} = B^i_j(i, j) \), where \( B^i_j(i, j) \) represents the probability in (5). Such rules are only created for neighboring nodes \( v_i, v_j \) that are connected by an edge. Suppose \( v_i \) represented a position of a sensitive asset like an embassy or a checkpoint. Using the approach presented earlier, a grammatical model \( G^*_T \) represents all trajectories with the target destination \( v_j \).

### Target Rendezvous

Consider the trajectories followed by two targets\(^1\) represented by position tracklets \( a^1_t \) and \( a^2_t \). A rendezvous is defined as an anomalous event where two targets meet at the same position \( v_i \) at time \( T \). Such a situation is depicted in Figure 7. The rendezvous of two targets at a given node \( v_i \) can be considered a destination-aware trajectory and modeled using the grammar rules described in the previous section. Define the multiple target position tracklet sequence \( a_T = [a^1_T, a^2_T] \) as a vector concatenation. The rendezvous of two targets is then modeled as a destination-aware trajectory \( a_{i_1}, ..., a_{i_k} \), \( a_T = [v_i, v_i]^T \) with the final state of both targets pinned to the intended meeting point \( v_i \) at time \( T \). The precomputed time-varying transition matrix \( B^i_j(i, j) \) is used in an extended state space model to define a higher-order (two-target) transition matrix

\[
B^{i,k}_{j,k}(m, n) = B^i_j(i, j) \otimes B^i_j(j, k),
\]

where \( i, j \in \mathcal{V}^m \) and \( m, n \in \mathcal{V}^m \times \mathcal{V}^m \). The \( \otimes \) operator represents the Kronecker product between two matrices. The notation \( B^{i,k}_{j,k}(m, n) \) represents the time-varying transition probability \( P \{ a_T = [a^1_T, a^2_T] = [v_i, v_i] \} \). The grammar model for target rendezvous has the same structure as destination-aware trajectory models. However, the nonterminal and terminal sets are different due to the expansion in the state space. The time-varying rule probabilities are specified in (6).

### Palindrome Paths

A palindrome refers to a sequence \( a_1, a_2, ..., a_p, a_p, ..., a_2, a_1 \) such that reversing the sequence results in the same pat-

\(^1\) The restriction to two targets is for simplification of notation; the model can be extended to an arbitrary number of targets. However, the state space increases exponentially with the number of targets.
LINEAR TRAJECTORIES

Linear trajectories are straight paths that are generated by constant velocity (CV) target dynamics obeying local Markov dependency. Linear grammar models are represented using the compact form $G^{CV} = \{ \alpha^t \}$, implying that the model can generate all trajectories involving $n$ movements of a target in the direction represented by the unit vector $\alpha$. A simple regular grammar for lines is characterized by rules of the form $S \rightarrow \alpha \bar{u} | \bar{u}$, with $\bar{u} \in \mathcal{V}^{vel}$ representing the target’s direction of motion.

ARCLIKE TRAJECTORIES

The compact form for arclike trajectories is $G^{arc} = \{ \alpha^t \bar{b}^t \bar{c}^t \}$, which is characterized by equal movements in opposing directions represented by the unit vectors $\bar{a}$ and $\bar{c}$ (Figure 9a). The notation $\bar{b}^t$ denotes an arbitrary number of movements in the direction represented by $\bar{b}$. A simple grammar capable of generating arcs of all lengths is shown in Figure 9b. We use the notion of arcs to represent U-turn and open trapezoidal patterns.

RECTANGLE TRAJECTORIES

We consider the modified-rectangle (m-rectangle) language (with associated grammar shown in Figure 10b) as $G^{m-rectangle} = \{ \alpha^t \bar{b}^t \bar{c}^t \bar{d}^t \}$. The m-rectangle grammar can model any trajectory comprising of four sides at right angles (not necessarily a closed curve) with at least two opposite sides being of equal length. The notations $\bar{b}^t$ and $\bar{d}^t$ represent an arbitrary number of movements in the corresponding directions represented by that mode. A full rectangle with both opposite sides of equal length cannot be modeled by an SCFG [24].

SUMMARY

In formal language theory, it is known (and provable using pumping lemmas [23]) that trajectories like arcs, rectangles, and palindromes are impossible to generate using Markov models exclusively [23]. In addition, the incorporation of reciprocal dynamics using time-varying rule probabilities allows destination-aware and rendezvous trajectories to be efficiently modeled. This section illustrated the use of such non-Markovian models in trajectory classification.

ILLUSTRATIVE EXAMPLE: MULTITARGET PATTERN OF LIFE CHANGE DETECTION

In this section, multiple targets are incorporated into anomaly detection as a change detection problem. The aggregate behavior of all targets moving in the surveillance space $\Lambda$ is assumed to arise from reciprocal dynamics characterized by the three-point transitions $Q(i,j,l)$. An anomaly can be defined as a change in the underlying reciprocal dynamics such that a normal regime $Q^{normal}(i,j,l)$ is in effect in the time interval $t = \{0, 1, \ldots, t-1\}$ and an abnormal regime $Q^{abnormal}(i,j,l)$ is in effect in the time interval $t = \{t, t+1, \ldots, T\}$. \(8\)
Spatiotemporal Trajectory Models

In [3], a surveillance application is described that seeks to detect changes in the pattern of life of a local population. Often, the local population is sympathetic toward insurgent and rebel groups. Information about anomalous events like the installation of an improvised explosive device (IED) can quickly propagate through the local population, which manifests itself as a sensitive asset located at node $v_i$. The normal operation of targets follows the behavior shown using dashed green lines. However, a hypothetical anomaly in the vicinity of node $v_i$ is detected by the solid red lines.

Change detection can be described as the problem of determining if or when, during the observation interval, the underlying reciprocal dynamics switches between two known models. We impose no prior knowledge on the switching time (we assume it is uniformly distributed). A pictorial depiction of such a scenario is shown in Figure 11. A sensitive asset is located at node $v_i$. The normal operation of targets follows the behavior shown using dashed green lines. However, a hypothetical anomaly in the vicinity of node $v_i$ is detected by the solid red lines.

Change detection can be described as the problem of determining if or when, during the observation interval, the underlying reciprocal dynamics switches between two known models. We impose no prior knowledge on the switching time (we assume it is uniformly distributed). A pictorial depiction of such a scenario is shown in Figure 11. A sensitive asset is located at node $v_i$. The normal operation of targets follows the behavior shown using dashed green lines. However, a hypothetical anomaly in the vicinity of node $v_i$ is detected by the solid red lines.

Change detection can be described as the problem of determining if or when, during the observation interval, the underlying reciprocal dynamics switches between two known models. We impose no prior knowledge on the switching time (we assume it is uniformly distributed). A pictorial depiction of such a scenario is shown in Figure 11. A sensitive asset is located at node $v_i$. The normal operation of targets follows the behavior shown using dashed green lines. However, a hypothetical anomaly in the vicinity of node $v_i$ is detected by the solid red lines.

Change detection can be described as the problem of determining if or when, during the observation interval, the underlying reciprocal dynamics switches between two known models. We impose no prior knowledge on the switching time (we assume it is uniformly distributed). A pictorial depiction of such a scenario is shown in Figure 11. A sensitive asset is located at node $v_i$. The normal operation of targets follows the behavior shown using dashed green lines. However, a hypothetical anomaly in the vicinity of node $v_i$ is detected by the solid red lines.

Change detection can be described as the problem of determining if or when, during the observation interval, the underlying reciprocal dynamics switches between two known models. We impose no prior knowledge on the switching time (we assume it is uniformly distributed). A pictorial depiction of such a scenario is shown in Figure 11. A sensitive asset is located at node $v_i$. The normal operation of targets follows the behavior shown using dashed green lines. However, a hypothetical anomaly in the vicinity of node $v_i$ is detected by the solid red lines.
can also be computed in a recursive manner for each target using the prefix probability in (11).

**NUMERICAL EXAMPLES**

In the next section, the classification performance of single-target trajectories is considered. The section on the pattern of life analysis explores the use of grammatical models toward rendezvous detection for two targets and change detection in aggregate target behavior for pattern of life analysis.

**SINGLE-TARGET SCENARIOS**

SCFG models are compared to equivalent HMMs whose parameters are learned by a Baum-Welch estimation procedure over synthetic tracklet sequences generated to obey its characteristic feature (such as its shape or destination). We demonstrate the effectiveness of our approach using receiver operating characteristic (ROC) curves for detection performance.

**Velocity Tracklet–Based Simulations**

A continuous-valued target state trajectory \( s_{0:n} \) is generated for three kinds of shapes: lines, arcs, and m-rectangles. For example, an arc (like that shown in Figure 1) can be generated by concatenating the trajectories generated by three CV models. The first CV model is initialized using a normal distribution that has a mean with the position at origin and a unit velocity in the northeast direction such that \( \mathbf{P}(s_0) \sim N(0, 0, \frac{1}{2}, \frac{1}{2}, \Sigma) \) The covariances matrix \( \Sigma \) can be chosen to generate trajectories with different signal-to-noise ratios. This model is then used to generate the upward part of the arc for \( t = 0, \ldots, t_1 \) time points. The second CV model is initialized using the final position of the target in CV model 1 and with velocity magnitude \( v_{\text{arc}}^n = \sqrt{(x_{\text{arc}}^n)^2 + (y_{\text{arc}}^n)^2} \) equal to the final velocity of the target in CV model 1. However, the velocity magnitude is concentrated in the z-axis such that \( \mathbf{P}(s_{t_1}) \sim N((x_{\text{arc}}^n, y_{\text{arc}}^n, v_{\text{arc}}^n, 0), \Sigma) \). Finally, the last part of the arc trajectory is created by running a third CV model for time points \( t = t_1 + 1, \ldots, T \) such that the initial state \( \mathbf{P}(s_{t_1}) \sim N((x_{\text{arc}}^n, y_{\text{arc}}^n, v_{\text{arc}}^n, 0), \Sigma) \).

In all simulations, we generate trajectories of a length of 1,000 time points at the fine timescale \( T \) with randomly chosen segment lengths obeying shape characteristics. Monte-Carlo runs of more than 1,000 trajectories are used for the results shown in Figure 2 and all subsequent simulations. A noisy radar observation process is used, and the measurements \( z \) are fed into an extended Kalman filter. Ideal detection is assumed, and the data association problem associated with multiple targets is not incorporated in the simulations. The filter estimates are then aggregated every 20 time points, and velocity tracklets \( \hat{v}_{ij} \) are generated at the slower timescale \( \tau \). For the shape-based trajectories, we create four types of models: \( G_{\text{line}}^{\text{shape}} = (G_{\text{line}}, G_{\text{line}}^n, G_{\text{line}}^{\text{rectangle}}, G_{\text{line}}^{\text{arc}}) \) and \( G_{\text{arc}}^{\text{shape}} = (G_{\text{arc}}, G_{\text{arc}}^n, G_{\text{arc}}^{\text{rectangle}}, G_{\text{arc}}^{\text{arc}}) \) using both SCFG and HMM frameworks. The \( G_{\text{arc}}^{\text{shape}} \) models for arbitrary trajectories are also added to each model set. The transition matrix of the arbitrary HMM is set up such that the transition probability from tracklet \( v_j \) to \( v_i \) is inversely proportional to the angular separation of the directions represented by \( v_j \) and \( v_i \). It can represent jumps from any movement direction to all other directions. Such a model is the archetype of a random walk. A similar model is also used in grammatical form for the arbitrary SCFG model. Equivalent HMMs for lines, arcs, and rectangles are created using a left-to-right transition structure, and the transition probabilities are learned from synthetically generated example sequences for each shape.

**Position Tracklet–Based Simulations**

Position-tracklet trajectories depend on the incorporation of the three-point transitions \( Q(i, j, l) \) described in the section on random walk models. Traffic authorities in various cities collect and store the turns ratio and traffic flow information over road networks using traffic cameras and induction loop sensors. Traffic flow \( \omega(i, j) \) is a count of vehicles traveling between two neighboring nodes \( i \) and \( j \). The traffic flow is undefined for nodes that are not connected by a road. The turns ratio \( \kappa(i, j, l) \) at node \( v_j \) represents the proportion of cars turning toward node \( v_j \) when arriving at node \( v_i \) from node \( v_j \). The three-point transitions \( \hat{Q}(i, j, l) \) can be empirically estimated using these quantities as

\[
\hat{Q}(i, j, l) = \frac{\omega(i, j) \cap \omega(j, l)}{\sum_j \omega(i, j) \cap \omega(j, l)} = \frac{\omega(i, j) \kappa(i, j, l)}{\sum_j \omega(i, j) \kappa(i, j, l)},
\]

where \( \omega(i, j) \cap \omega(j, l) \) denotes the number of vehicles that travel first from node \( v_i \) to node \( v_j \) and finally on to node \( v_l \). However, this quantity cannot be computed without specifically identifying each vehicle. Consequently, the approximation in (9) is used. In the absence of traffic data, auxiliary information from cellular localization or vehicular global positioning system traces can also be used to estimate \( Q(i, j, l) \). In such cases,
the estimation does not require the approximation in (9), because each cell phone or vehicle can be uniquely identified. However, the traces must be quantized to grid positions in the surveillance space. For the purposes of simulations, a synthetic three-point transition matrix is used such that the three-point transitions \(Q(i, j, l)\) at each node \(v_i\) are inversely proportional to the distance between \(v_i\) and destination node \(v_k\). Such a choice biases the target to make transitions to reach the destination \(v_k\) in the shortest number of hops.

**Destination-Aware Trajectories**

Using the three-point transitions described earlier and the time-varying transitions in (5), a destination-aware trajectory model \(G^{\text{st}}\) is created for an arbitrary node \(v_i\), which was chosen as an interesting node hypothetically containing a sensitive asset. We then simulate multiple trajectories of length \(T\) with node \(v_i\) as the destination. These trajectories are observed using a noisy process such that

\[
P(\hat{a}_r = v_j | a_r = v_i) = \begin{cases} 
0.8, & \text{for } v_i = v_j \\
0.2, & \text{for } v_i \in \mathcal{N}_j,
\end{cases} \tag{10}
\]

where \(\mathcal{N}_j\) refers to the set of intersections connected to node \(v_j\) by a street and \(|\mathcal{N}_j|\) refers to the number of connected nodes. This noise distribution was chosen to incorporate averaging and quantization effects from the tracklet estimator in (2). As before, models for both SCFG and HMM frameworks are created such that \(G^{\text{st}} = \{G^{\text{st}}_1, G^{\text{st}}_2, \ldots, G^{\text{st}}_v\}\) and \(G^{\text{st}}_{\text{rand}} = \{G^{\text{st}}_{\text{rand}}_1, G^{\text{st}}_{\text{rand}}_2, \ldots, G^{\text{st}}_{\text{rand}}_u\}\). The \(G_{\text{st}}\) models are destination constrained but have a final destination at some node \(i = k\). We chose several such nodes within 2–3 hops of the intended destination \(k\). The detection performance can be seen in Figure 13a.

**Palindrome Trajectories**

Examples are also provided for the detection of palindrome paths over the road network \(\Lambda\). A palindrome trajectory is artificially simulated by first generating an arbitrary trajectory for \([T/2]\) time points and then appending the reversed sequence to create a retraced path. The same noisy observation process as in (10) is used to obtain tracklet measurements \(\tilde{a}_{ir}\). We use a destination-aware model with the same start \(a_0 = v_x\) and end \(a_T = v_x\) points represented as \(G^{\text{st}}_{vx}\), as well as a more general SCFG model allowing random transitions like a random walk within the model set \(G^{\text{st}} = \{G^{\text{st}}_1, G^{\text{st}}_2, \ldots, G^{\text{st}}_v\}\). The equivalent HMM has a fully connected state transition diagram, and the rule probabilities are learned using Baum-Welch reestimation [25] from example palindrome trajectories generated from its generative SCFG model. The performance results are shown in Figure 13b.

The target rendezvous event can be considered an expanded state space version of the destination-aware trajectory model. Simulations show similar results to the single-target destination-aware case. As a result, simulation results are not shown for such events.

**PATTERN OF LIFE ANALYSIS**

The multitarget change detection problem presented in the section with the illustrative example is examined in this section. The normal operating regime consists of a regular 50-node network, as shown in Figure 2b. The normal regime is simulated by making the three-point transitions \(Q(i, j, l)\) inversely proportional to the distance between the node \(v_i\) and the destination node \(v_k\) to bias the target to reach the destination as quickly as possible. An anomalous regime is then created by remov-
like movement patterns and quantized positions are used as syntactic subunits within an SCFG framework to perform anomaly detection. The expressive power of SCFG models is able to capture long-term dependencies in intent-driven trajectories while utilizing efficient polynomial time algorithms for their inference. The Earley-Stolcke parser is modified using a scaling trick to allow processing of long tractable sequences. A novel interpretation of reciprocal processes using time-varying SCFG rules is also presented. We devised novel models for anomalous events like target rendezvous and palindrome paths. Furthermore, the pattern-of-like-analysis problem was formulated as a change detection problem using multiple target trajectories. Finally, a comprehensive numerical evaluation of our proposed models was carried out to show an increase in the detection performance over conventional Markov models.

APPENDIX A: TUTORIAL OVERVIEW OF SCFGS

STRUCTURAL DESCRIPTION OF SCFG

A grammar \( G = (X, \mathcal{V}, S, \mathcal{R}) \) is a quadruple consisting of a set \( X \) of latent variables \( X_i, i = 1, \ldots, |X| \) called nonterminals; a set \( \mathcal{V} \) of discrete observations \( v_i, i = 1, \ldots, |\mathcal{V}| \) called terminals; a special start symbol \( S \); and a set \( \mathcal{R} \) of production rules \( r_m, m = 1, \ldots, |\mathcal{R}| \). In the context of trajectory modeling, nonterminals represent hierarchical structural segments of a trajectory while terminals represent actual subunits of a trajectory. The production rules \( r_m \) describe the manner in which the trajectory can evolve by combining the hidden structural parts of the trajectory. Each nonterminal \( X_i \) can have \( n_{x_i} \) alternative rules that can be chosen when nonterminal \( X_i \) is being considered for rewriting.

A context-free grammar \( G^{CFG} \) additionally has the property that its production rules can only have the form \( X_i \rightarrow \alpha \), such that a nonterminal \( X_i \) can be rewritten as an arbitrary string \( \alpha \epsilon (X \cup \mathcal{V})^* \). The notation \( (X \cup \mathcal{V})^* \) denotes an arbitrary (possible empty) combination of nonterminal and terminal symbols. An SCFG \( G^{SCFG}(G^{CFG}, \mathcal{P}) \) is a pair consisting of a context-free grammar and a rule probability set \( \mathcal{P} = \{ p_r \}_{r \in \mathcal{R}} \) assigning a probability \( p_r \) to each rule \( r \in \mathcal{R} \). The rule probabilities create a conditional probability distribution over all alternative expansions of a nonterminal \( X_i \) such that \( \sum_{r \in R} p_r = 1.0 \) for each \( X_i \).

In the following discussion, an SCFG is treated as a triple stochastic process with two layers of latent variables. The rule choices used in the derivation of a sentence is the first latent layer. A graphical representation of the sequence of rewriting rules used until epoch \( t \) is called a partial parse tree \( \mathcal{P} \). The sequence of rule choices is also called a derivation that generates a sequence of “clean” terminal symbols \( a_{\mathcal{P}} \) on termination. The clean terminal symbols represent the second hidden layer because they are only observed in noise. The final observation process generates a sequence \( \hat{a}_t \) of “noisy” terminals. The SCFG measurements are represented by an observation distribution \( C(i, j) = P(\hat{a}_t = v_j | a_t = v_i) \). The noisy observation layer can be subsumed into the grammar rules by creating a special “emitting” nonterminal for each terminal. However, this
increases the size of the grammar and makes the rules less intuitive [26].

EXAMPLE OF SCFG PATTERN GENERATION

To provide more intuition, we compare and contrast SCFGs to HMMs. Consider the grammar shown in Figure 15a. This is a simple grammar generating palindromes over a binary-valued alphabet $\mathcal{V} = \{a, b\}$. The SCFG-generation process starts with a special start symbol $S$. Initially, only nonterminal $S$ is present in the parse tree shown in Figure 15a. We then sample one of the rules from the conditional distribution over all alternative rewrite rules of the start symbol $S$. The next generation of the tree is created by replacing the nonterminal under expansion (the start symbol $S$ in this case) with the right-hand side of the sampled rewrite rule. In the example, rule $r_1$ was chosen with probability $p_1$, and the start symbol $S$ is replaced by the right-hand side “$a S a$” of rule $r_1$. The procedure is repeated in each generation by leftmost derivation in which only the leftmost nonterminal in the current symbol stack is rewritten with the right side of a chosen production rule. The resultant parse tree $\tilde{\mathcal{P}}$ is depicted on the right in Figure 15a. At the end of the derivation process, no more nonterminal symbols exist and the symbol stack only contains clean terminal symbols. The noisy observation process is represented by associating a discrete probability mass function $C(i, j) = P(\xi_i|\nu_j)$ to each clean terminal $\xi_i$. The right bottom part of Figure 15a depicts the noisy observation process. The hat accent is used to differentiate noisy terminals $\hat{\alpha}$ from clean terminals $\alpha$.

An HMM attempting to generate palindromes is also shown in Figure 15b. Markov models cannot exclusively generate palindromes [23]; hence, the model in Figure 15b forms a poor generative model for palindromes. Nevertheless, it is instruc-
forms a linear directed graph, the SCFG parse tree exhibits a branching structure representing self-embedding rules of the form $S \rightarrow a \ S \ a$. These rules are the basis for the ability of SCFGs to model long-term dependencies in a symbol sequence. SCFGs have been studied in probability theory under the umbrella of random branching processes. In particular, they form a special class of branching processes called multitype Galton-Watson branching processes [27]. From the comparison in Figure 15, it can also be observed that the number of parameters in the HMM is larger than the number in the analogous SCFG model. The predictive entropy of an SCFG has been shown in [28] to be greater than a comparative HMM with a similar number of parameters.

INFECTION USING SCFGS

In this article, we are mainly concerned with (a) computing trajectory likelihoods $P(\tilde{a}_t; G^{SCFG})$ and (b) recovering the most likely clean terminal sequence $\arg \max_{a_{tT}} P(\tilde{a}_{tT} | a_{tT}; G^{SCFG})$ from an observed sentence $\tilde{a}_{tT}$ given an SCFG model $G^{SCFG}$. The model likelihoods are used to classify anomalous patterns as formulated in (3) and in the change detection problem in the section with the illustrative example. The most likely parse tree estimation is used to recover the clean terminal sequence originally generated by the SCFG. As described in the introduction, the Earley-Stolcke parser is used in this article due to its ability to deal with unrestricted context-free rules. Moreover, it is capable of generating likelihoods for partial sentences due to its incremental left-to-right operation.

The Earley-Stolcke parser builds leftmost derivations of a string by keeping track of all possible derivations that are consistent with the observations up to a certain epoch $t$. As more observations are revealed, the set of possible derivations (each of which corresponds to a partial parse tree) can either expand as new choices are introduced or shrink as a result of resolved ambiguities. As each observation $\tilde{a}_t$ is received by the parser, a set of states is created that represents all derivations that can explain the observations $\tilde{a}_t$. An Earley state is the basic control structure used in the operations of the Earley-Stolcke parser. Each state $u_t \in U$ represents a rewrite rule $r_{u_t} \in R$ such that the observation $\tilde{a}_t$ is derived from its right-hand side. An Earley state is denoted by $\tilde{a}_t \rightarrow \lambda : Y \tilde{a}_t$. The uppercase letters $X$ and $Y$ are nonterminals, and $\lambda$ and $\mu$ are substrings of nonterminals and terminals. The index $t$ represents the current epoch. The dot "\cdot" demarcates the portion of the right-hand side that has already been recognized by the parser. The index $t$ is a backpointer to
the epoch at which the parser rewrote nonterminal $X$ using an instance of production rule $r_{\rightarrow X}$. Each state is a link in an incomplete chain of rule choices that could have generated the observed sequence $\hat{a}_{u_t}$.

Each state is also associated with a forward probability $\alpha$ and an inner probability $\gamma$. The forward probability $\alpha(u_t)$ of a state $u_t$ represents the probability of the grammar $G^{SCFG}$ deriving the observations $\hat{a}_{u_t}$, while passing through the state $u_t = \omega_t X \rightarrow \lambda : Y \mu$ at epoch $t$. This is analogous to the definition of forward probability for the case of an HMM $G^{HMM}$ [25]. The inner probability $\gamma(u_t)$ of a state $u_t$ is defined as the probability of a particular rule generating an infix of the sentence $\hat{a}_{u_t}$ represented by the accumulated probabilities of all derivations starting with a state $\omega_t X \rightarrow \lambda : Y \mu$ and ending at the state $\omega_{t'} X \rightarrow \lambda Y \mu$.

For the purposes of dealing with the start symbol, the Earley-Stolcke parser is initialized with a dummy state $\omega_0 \rightarrow S[ \alpha = 1, \gamma = 1]$. At each epoch, the states in an Earley set $u_t$ are processed in order by performing one of three operations on each state. These operations may add more states to $u_t$ and may put states in a new state set $u_{t+1}$. Whenever an operation attempts to add a new state, it is linked to an existing state. Such a sequence of linked states represents different rules choices that could have generated the input string:

1) The prediction operation is applied to states when there is a nonterminal to the right of the dot. It causes the addition of one new state to $u_t$ for each alternative production rule of that nonterminal. The dot is placed at the beginning of the production rule in each new state. The backpointer is set to $u_t$. Thus, the predictor adds to $u_t$ all productions that might generate substrings beginning with $a_{\alpha}$. More formally, for a state $\omega_{t'} X \rightarrow \lambda : Y \mu$ in the state set $u_t$, the predictor adds a new predicted state $\omega_{t''} Y \rightarrow v$ for each of the alternative production rules $(Y \rightarrow v) \in \mathcal{R}$. A link is thus created between these states.

2) The scanning operation is applicable only when there is a terminal $a$ to the right of the dot. The scanner compares that symbol $a$ with the observation $\hat{a}_{\omega_t}$, and if they match, it adds the state to $u_{\alpha+1}$ with the dot moved over one symbol in the state to indicate that the terminal symbol has been scanned. If $\omega_{t'} X \rightarrow \lambda : a \mu$ exists and $P(\hat{a}_{\omega_t} | a) > 0$, the scanning operation adds a new scanned state $\omega_{\alpha+1} X \rightarrow \lambda a : \mu$ to state set $u_{\alpha+1}$. The symbol likelihood $P(\hat{a}_{\omega_t} | a) > 0$ allows us to incorporate a noisy observation process into SCFG generation.

3) The completion operation is applicable to a state if its dot is at the end ($\omega_{\alpha+1} Y \rightarrow v$) of its production rule. Such a state is called a complete state. For every complete state, the parser backtracks to the state set $u_t$ indicated by the pointer in the complete state and adds all states from $u_t$ to $u_{\alpha}$ that have $Y$ (the nonterminal corresponding to that production) to the right of the dot. It moves the dot over $Y$ in such states. Intuitively, $u_t$ is the state set where $Y$ was first expanded. The parser has seen evidence $\hat{a}_{\omega_t}$ that $Y \rightarrow v$ was used, so we move the dot over the $Y$ in these states to show that its terminal yield been successfully scanned. A completion operation adds a new state $\omega_{\alpha+1} X \rightarrow \lambda Y \mu$ (called a completed state) using $\omega_{\alpha+1} X \rightarrow \lambda : Y \mu$ and $\omega_{\alpha+1} Y \rightarrow v$, where $\alpha \leq \alpha'<\alpha+1$.

The Earley-Stolcke parser continues in this manner until all observation $\hat{a}_{u_t}$ symbols have been scanned. If the final state set $u_{\alpha}$ contains the state $\omega_{\alpha} S' \rightarrow S$, then the algorithm terminates successfully. It represents a successful parse of the sentence $a_1, \ldots, a_{\alpha-1}, a_{\alpha}$. The recursive updates of the forward probability $\gamma$ and inner probability $\alpha$ for each of the state operations are summarized in Algorithm 1.

To deal with underflow issues, state-independent scaling must be applied at every epoch. Define the prefix probability $\xi_t = P(\hat{a}_{u_t} | G^{SCFG})$ as the probability of the grammar $G^{SCFG}$ generating the observations $\hat{a}_{\omega_t}, \ldots, \hat{a}_{\omega_{\alpha}}$ as the prefix of a complete sentence $\hat{a}_{\omega_{\alpha}}$. Philosophically, this involves a summation over all possible suffixes $\beta \in (X \cup \mathcal{V})^*$. However, in [29], it is shown that this is equivalent to the sum of the forward probabilities $\alpha_t$ of all scanned states in the state set $u_t$ such that

$$\xi_t = \sum_{u_t \in \mathcal{U}} P(\hat{a}_{u_t}) = \sum_{u_t \in \mathcal{U}} \alpha_t X \rightarrow Y \mu, \gamma$$

where $n$ is a scanned state. An appropriate choice for the scaling factor $c_i$ in each state set is to use the inverse of the one-step prediction probability defined as

$$c_i = \frac{\xi_{i-1}}{\xi_i}$$

The scaling factor $c_{\omega_0} = 1/\xi_{\omega_0}$ is initialized using the prefix probability of the first observation $\hat{a}_{\omega_0}$. The scaling factor at all other epochs $i = 1, \ldots, T$ is computed as $c_i = 1/\xi_{\omega_t}$. The forward $\alpha$ and $\gamma$ probabilities of all scanned states are multiplied by the scaling factor to prevent underflow issues. The likelihood

$$L_i = P(\hat{a}_{\omega_t} | G^{SCFG})$$

of the observation sequence $\hat{a}_{\omega_t}$ can be computed from the sequence of scaling factors $c_{\omega_0}, \ldots, c_i$ as

$$L_i = \frac{1}{\prod_{\alpha \leq \alpha'} c_{\omega_{\alpha'}}}.$$
\[
\gamma' = \gamma \mathcal{P}(\hat{A}_t | a)
\]

\[
\zeta_t = \sum_{x \in A} \alpha_t(x, \rightarrow \lambda a \cdot \mu)
\]

7. Compute \( c_i = \frac{1}{\zeta_{t-1}} \) by computing \( \zeta_{t-1} \), from (12)

8. Scale \( \alpha, \gamma \) for all scanned states using \( c_i \)

9. \( \text{Completion} \)
   
   \[ \text{for } j' | Y \rightarrow v \cdot [\alpha', \gamma'] \in u_d, \text{ do} \]
   
   \[ \text{for } j' | X \rightarrow \lambda \cdot Z \mu(\alpha, \gamma) \in u_d, \text{ do} \]
   
   \[ \text{if } R_j(Z, Y) \neq 0 \text{ then} \]
   
   \[ \text{Add } j' | X \rightarrow \lambda Z \mu(\alpha', \gamma') \]
   
   \[ \alpha' + = \alpha y' R_j(Z, Y) \]
   
   \[ \gamma' + = \gamma y' R_j(Z, Y) \]

   \[ \text{Prediction} \]

10. \( \text{for } j' | Y \rightarrow v \cdot [\alpha', \gamma'] \in u_d, \text{ do} \)

11. \( \text{Add } j' | Y \rightarrow v \cdot [\alpha', \gamma'] \) if \( R_j(Z, Y) \neq 0 \).

12. \( \alpha' + = \alpha R_j(Z, Y) \mathcal{P}(Y \rightarrow v) \)

13. \( \gamma' + = \gamma R_j(Z, Y) \)

14. \( \text{return } u_d, c \)

In addition to sequence likelihoods \( L_y \), we are interested in recovering the sequence of clean tracklets \( a_{0:y} \). For an SCFG model, this involves computing the Viterbi parse of the observations \( \hat{a}_{0:y} \). A Viterbi parse recovers the sequence of rule choices (or the parse tree \( \mathcal{P} \)) that assigns maximum probability to \( \hat{a}_{0:y} \), among all possible parses. The same sequence of operations as described in Algorithm 1 can be used to recover the Viterbi parse. However, each state \( u_i^n \) also keeps track of its Viterbi path probability \( \phi \). The Viterbi probability \( \phi \) is propagated in the same manner as the inner probability \( \gamma \). During completion, the accumulated sum \( + = \) is replaced by a maximum over all products that contribute to the completed state. The complete state \( j' | Y \rightarrow v \), at the current epoch \( t \) associated with the maximum is recorded as the predecessor of \( j' | X \rightarrow \lambda Z \cdot \mu \). The Viterbi probability update also does not use the collapsing loop factor \( R_j(Z, Y) \), because loops can only lower the probability of a derivation. Once the final state \( 3 | S \rightarrow S \) is accepted by the parser, a recursive procedure can backtrack over the Earley state sets to recover the Viterbi parse \( \mathcal{P}^* \) [29]. The leaf nodes of the parse tree can then be traversed to recover the clean terminals \( a_{0:y} \).

### Example of Parser Operation

The parser operations are illustrated with the use of a simple yet nontrivial example utilizing the palindrome grammar in Figure 15a. The grammar has four rules with the rule probabilities \( p_1 = 0.4, p_2 = 0.2, p_3 = 0.1 \), and \( p_4 = 0.3 \). Suppose that we are presented with the query pattern \( a b a \). The classification results are shown in Table 1.

<table>
<thead>
<tr>
<th>State set ( t = 0 )</th>
<th>( \text{Operations} )</th>
<th>( \text{State} )</th>
<th>( \text{Attributes} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0 ) ( \rightarrow S )</td>
<td>Predicted</td>
<td>( 0 ) ( \rightarrow aSa )</td>
<td>( \alpha = 1.0, \gamma = 0.4 )</td>
</tr>
<tr>
<td>( 0 ) ( \rightarrow aSa )</td>
<td>Predicted</td>
<td>( 0 ) ( \rightarrow bSb )</td>
<td>( \alpha = 0.2, \gamma = 0.2 )</td>
</tr>
<tr>
<td>( 0 ) ( \rightarrow a )</td>
<td>Predicted</td>
<td>( 0 ) ( \rightarrow b )</td>
<td>( \alpha = 0.3, \gamma = 0.3 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State set ( t = 1 )</th>
<th>( \text{Operations} )</th>
<th>( \text{State} )</th>
<th>( \text{Attributes} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1 ) ( \rightarrow S )</td>
<td>Scanned</td>
<td>( 1 ) ( \rightarrow a\cdot Sa )</td>
<td>( \alpha = 0.36, \gamma = 0.36 )</td>
</tr>
<tr>
<td>( 1 ) ( \rightarrow b\cdot Sb )</td>
<td>Predicted</td>
<td>( 1 ) ( \rightarrow a )</td>
<td>( \alpha = 0.09, \gamma = 0.09 )</td>
</tr>
<tr>
<td>( 1 ) ( \rightarrow b )</td>
<td>Predicted</td>
<td>( 1 ) ( \rightarrow S )</td>
<td>( \alpha = 0.03, \gamma = 0.03 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State set ( t = 2 )</th>
<th>( \text{Operations} )</th>
<th>( \text{State} )</th>
<th>( \text{Attributes} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 2 ) ( \rightarrow S )</td>
<td>Scanned</td>
<td>( 2 ) ( \rightarrow a\cdot Sa )</td>
<td>( \alpha = 0.0152, \gamma = 0.04 )</td>
</tr>
<tr>
<td>( 2 ) ( \rightarrow a\cdot Sb )</td>
<td>Predicted</td>
<td>( 2 ) ( \rightarrow a )</td>
<td>( \alpha = 0.0038, \gamma = 0.01 )</td>
</tr>
<tr>
<td>( 2 ) ( \rightarrow b )</td>
<td>Predicted</td>
<td>( 2 ) ( \rightarrow S )</td>
<td>( \alpha = 0.1026, \gamma = 0.27 )</td>
</tr>
<tr>
<td>( 2 ) ( \rightarrow b )</td>
<td>Predicted</td>
<td>( 2 ) ( \rightarrow S )</td>
<td>( \alpha = 0.1008, \gamma = 0.0108 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State set ( t = 3 )</th>
<th>( \text{Operations} )</th>
<th>( \text{State} )</th>
<th>( \text{Attributes} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 3 ) ( \rightarrow S )</td>
<td>Scanned</td>
<td>( 3 ) ( \rightarrow a\cdot Sa )</td>
<td>( \alpha = 0.03344, \gamma = 0.4 )</td>
</tr>
<tr>
<td>( 3 ) ( \rightarrow b\cdot Sb )</td>
<td>Predicted</td>
<td>( 3 ) ( \rightarrow a )</td>
<td>( \alpha = 0.00836, \gamma = 0.1 )</td>
</tr>
<tr>
<td>( 3 ) ( \rightarrow b )</td>
<td>Predicted</td>
<td>( 3 ) ( \rightarrow S )</td>
<td>( \alpha = 0.02508, \gamma = 0.03 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State set ( t = 4 )</th>
<th>( \text{Operations} )</th>
<th>( \text{State} )</th>
<th>( \text{Attributes} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 4 ) ( \rightarrow S )</td>
<td>Predicted</td>
<td>( 4 ) ( \rightarrow S )</td>
<td>( \alpha = 0.0056, \gamma = 0.0056 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \text{Completed} )</th>
<th>( \text{Operations} )</th>
<th>( \text{State} )</th>
<th>( \text{Attributes} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 5 ) ( \rightarrow aSa )</td>
<td>( \alpha = 0.09072, \gamma = 0.09072 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 5 ) ( \rightarrow aSb )</td>
<td>( \alpha = 0.00056, \gamma = 0.00056 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 5 ) ( \rightarrow bSa )</td>
<td>( \alpha = 0.00056, \gamma = 0.00056 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 5 ) ( \rightarrow bSb )</td>
<td>( \alpha = 0.0056, \gamma = 0.0056 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 5 ) ( \rightarrow b )</td>
<td>( \alpha = 0.002508, \gamma = 0.002508 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{Completed} )</td>
<td>( \text{Operations} )</td>
<td>( \text{State} )</td>
<td>( \text{Attributes} )</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td>( 6 ) ( \rightarrow S )</td>
<td>( \alpha = 0.008208, \gamma = 0.0216 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 6 ) ( \rightarrow S )</td>
<td>( \alpha = 0.007524, \gamma = 0.09128 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 6 ) ( \rightarrow S )</td>
<td>( \alpha = 0.007524, \gamma = 0.09128 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The palindrome grammar of Figure 15a can generate the string \(aba\) using four derivation paths. Each path results from a different sequence of applied probability rules or because of nonideal detection probabilities of the terminal symbols. The sentence probability of the input pattern \(aba\) is the sum of all possible derivation paths. Due to ambiguity in the grammar-generation process, the sentence probability must account for all possible ways in which a particular terminal sequence can be generated. As can be seen from the parser tables in Table 1, the completed dummy state \(\bar{S} \rightarrow S\) has the correct forward probability of 0.09128 obtained independently by summing the individual probabilities of each alternative derivation path. The numerical superscript over the arrows in the figure represents the conditional probability of choosing that particular rule expansion when a nonterminal is being expanded.

**REFERENCES**


