Processing and Analysis of Position Data Streams

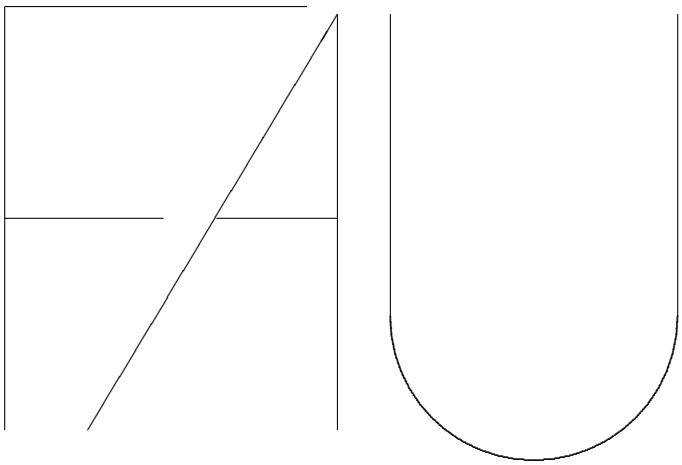
Masterarbeit

Johannes C. Tenschert

Lehrstuhl für Informatik 6
(Datenmanagement)

Department Informatik
Technische Fakultät

Friedrich Alexander-
Universität
Erlangen-Nürnberg



Processing and Analysis of Position Data Streams

Masterarbeit im Fach Informatik

vorgelegt von

Johannes C. Tenschert

geb. 01.05.1990 in Neumarkt i.d.OPf.

angefertigt am

Department Informatik
Lehrstuhl für Informatik 6 (Datenmanagement)
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Univ.-Prof. Dr.-Ing. habil. Klaus Meyer-Wegener
Dipl.-Inf. Sebastian Herbst

Beginn der Arbeit: 01.03.2014
Abgabe der Arbeit: 03.09.2014

Erklärung zur Selbständigkeit

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass diese Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Der Universität Erlangen-Nürnberg, vertreten durch den Lehrstuhl für Informatik 6 (Datenmanagement), wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Masterarbeit einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt.

Erlangen, den 03.09.2014

(Johannes C. Tenschert)

Kurzfassung

Verarbeitung und Analyse von Positionsdatenströmen

In der DFG Forschergruppe 1508 (BATS) wird ein Sensornetzwerk bestehend aus bodengestützten sowie mobilen Sensoren zur Überwachung von Fledermäusen entwickelt. In dieser Arbeit werden Techniken and Notationen herausgearbeitet, um Positionsdatenströme zu verarbeiten und zu analysieren.

Events und Aktivitäten sind Abstraktionen von Positionsdaten. Events werden unterteilt in Low- und High-Level-Events. Ein Low-Level-Event ist eine kontinuierliche Anfrage, deren Ergebnis als Event interpretiert wird. Ein High-Level-Event ist ein (komplexes) Muster von anderen Events. Der Ansatz verbindet damit Datenstromverarbeitung und Complex Event Processing. Aktivitäten werden in einer erweiterten Form von UML-Zustandsdiagrammen modelliert.

Die Definition von Aktivitäten erlaubt es, Verhalten vorherzusagen. Ein Verfahren für adaptive Vorhersagen zur Verbesserung der Genauigkeit von Vorhersagen wird vorgestellt für den Fall, dass mehrere Hypothesen für Abhängigkeiten in Bayesschen Netzen vorliegen.

Von Biologen gewünschte bisherige Analysen wie Heatmaps und Local Convex Hulls wurden ebenso in den Prototypen integriert wie neue Analysen. Techniken, die zur Implementierung von Vorhersagen eingeführt wurden, werden ebenso zur Inferenz über bedingten Wahrscheinlichkeitsverteilungen angewendet. Zusätzlich können auch Eigenschaften wie die Dauer oder Region von Aktivitäten durch spezielle Tabellen mit bedingten Verteilungen untersucht werden. Darüber hinaus wurde ein Ansatz zur Definition von Persönlichkeit anhand dieser Tabellen vorgestellt.

Weiterhin wird der Einfluss von Datenqualität auf Ergebnisse von Analysen diskutiert und die Zuverlässigkeit insbesondere von adaptiven Vorhersagen sowie die Ausdrucksmächtigkeit vorgestellter Notationen evaluiert.

Abstract

Processing and Analysis of Position Data Streams

The DFG research group 1508 (BATS) develops a sensor network consisting of static and mobile sensors to monitor bats. This thesis elaborates techniques and notations to process and analyze position data streams.

Abstractions for position data were elaborated: events and activities. Events are further divided into continuous queries interpreted as events and event patterns. Hence, the approach combines data stream processing and complex event processing. UML state charts were extended to be suitable for modeling activities.

The definition of activities exploits Dynamic Bayesian networks to predict behavior. Adaptive predictions are proposed to increase precision of predictions if several hypotheses of dependencies are specified.

Several state-of-the-art analyses, e.g. heatmaps and local convex hulls, as well as new analyses are implemented in the prototype. Techniques to predict activities additionally support inference on available Conditional Probability Tables as well as Conditional Tables for characteristics of activities. A possible definition of personality of animals is proposed.

Finally, the approach and prototype is evaluated against the objectives as well as reliability of predictions, influence of data quality on analyses and expressiveness of presented notations.

Table of Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

CEP	Complex Event Processing
CPT	Conditional Probability Table
DBMS	Database Management System
DBN	Dynamic Bayesian Network
DBS	Database System
DSMS	Data Stream Management System
DSS	Data Stream System
EPL	Event Processing Language
HMM	Hidden Markov Model
IPC	Inter Process Communication
LoCoH	Local Convex Hull
MCP	Minimum Convex Polygon
RPC	Remote Procedure Call

1 Introduction

I can't be as confident about computer science as I can about biology. Biology easily has 500 years of exciting problems to work on. It's at that level.

Donald Knuth (1993)

Two important goals of computer science are to automatize the acquisition of information and to enable access to information, especially to support solving problems in other disciplines. As Donald Knuth pointed out many years ago, biology has many exciting problems to work on, and exploring the behavior of bats is definitely one of them. The following chapters present techniques and a prototype to support biologists on analyzing position data streams to gain knowledge on how and why bats and other animals behave the way they do.

1.1 Motivation

In recent years, technological advances as well as the everlasting miniaturization of integrated circuits and their costs allowed to deploy sensor nodes in more and more circumstances. It is now technically and economically viable to attach sensors to animals as small and light-weighted as bats. There is a substantial amount of questions to be answered on the behavior of bats.

Currently, their position data are captured per pedes or via a vehicle. Automatically gathering and analyzing positions significantly increases the possible resolution, amount and coverage of collected data. Moreover, it leads to substantial reductions of manual work and allows biologists to focus more on interpreting the results of state-of-the-art and novel analyses.

The analysis of position data streams and their aggregation to behavior and activities raises several interesting problems. One important aspect of such an analysis is how to interpret incoming position data and how to include knowledge of domain experts. The abstraction level of movement descriptions may vary, for example ranging from very concrete behavior like a movement track in a certain shape to the abstract behavior *flying*.

Nonetheless should those descriptions influence the detection and aggregation of behavior. Another crucial aspect is how the analysis of position data streams could benefit both from data stream and complex event processing.

Since animals are not the only entities being tracked in this day and age, there is a wide range of possibilities of applying the results of this work.

1.2 Scenario

This thesis was created in the context of the DFG research group 1508 or rather BATS as it is called in the whole document. Its primary goal is "to gain fundamental insight in the design, construction and operation of resource-scarce, heterogenous, intelligent networks consisting of static and mobile sensors" [DFG13]. Biologists, computer scientists and electrical engineers collaborate to develop and demonstrate methods and techniques for gathering data, data management and data analysis. The scope of this thesis especially covers topics of data analysis and data management.

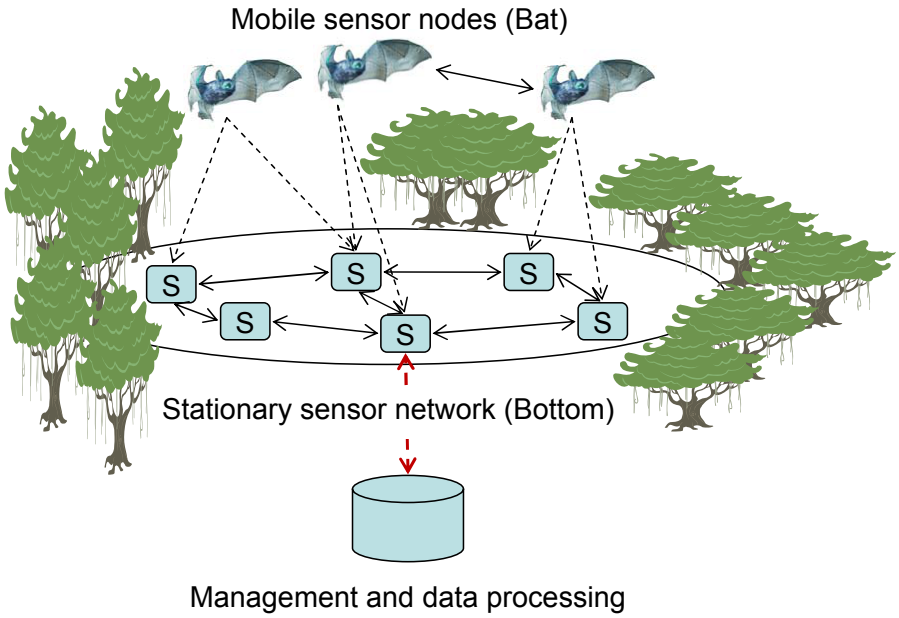


Figure 1.1: Topology and Architecture of the Heterogenous Sensor Network [DFG13]

Figure 1.1 outlines the architecture of the sensor network. Bats are equipped with mobile sensor nodes. Those mobile sensor nodes are monitored by a network of ground-based, static sensors gathering streams of positions and environmental data.

Even though the sensor network is not yet deployed, there are some assumptions on the available sensors and data. Positions of bats are emitted every second. Each position element contains a value for x, y and z. The system will achieve a precision of x and y in the range of meters or at worst tens of meters. Whether the z-axis is usable at all has to be evaluated in the field. Moreover, sensors can be attached to ground nodes. Hence, data input of sensors measuring temperature, humidity, brightness, noise and other environmental data is expected.

The design and implementation of methods, techniques and a finally deployed system are driven by both biologists questions and technological proposals to be verified and possibly revised. Biologists want to gain more insight into bat behavior. Moreover, they would like to use the developed system and its techniques to examine a broad range of species in the future. On the engineering side, there are many design decisions to be determined in order to create a wireless sensor network and the techniques to process gathered data. Those issues are not limited to data management, but also involve software infrastructure, network topology and the actual hardware to be deployed. BATS is divided into several subprojects[1] tackling those issues and elaborating biologists requirements.

1.3 Objectives

Processing position data streams in the context of tracking animals in sensor networks raises several interesting problems. Most of the challenges arise not only restricted to bats, but for a wide range of different species.

A crucial goal of this thesis is to describe a system to process position data streams into more expedient output to answer biologists questions. The approach to build such a system will be described in section 1.4. The objectives and description of the system are provided in the following subsections.

1.3.1 Questions in Biology

Biologists have several assumptions on the behavior of bats and hypotheses on probable behavior. They want to find out if those hypotheses are true and whether current theories on the behavior have to be revised. Moreover, they want to find new behavioral patterns

1 `http://www.for-bats.org/subprojects.shtml`

and gain more in-depth knowledge on existing ones. Current questions and behavioral hypotheses to be validated are elaborated in chapter 2.

Since the prototype is intended to be used by biologists, it should provide certain techniques to support experiments, for example by automatically calling external scripts or programs at the occurence of particular events. The integration of state-of-the-art analyses in the chosen processing model has to be assessed and executed.

As soon as the presented techniques are tested in practice, the list of questions to be answered will inevitably grow. Hence, it should be easy to reuse existing and to add new analyses.

1.3.2 Events and Activities

Events and activities of tracked animals or objects are an important result of the analysis of position data. They are an integral part of the animals behavior. Hence, events and activities are one crucial output of the prototype. Since their description depends on knowledge of domain experts, modeling plays an important role. As soon as activities are available, predictions of those activities get more and more interesting. Predictions may support experiments and are useful to validate the current state of behavioral knowledge.

1.3.3 Visualization

In order to support biologists to interpret findings, visualizations have to be provided. Examples for reasonable visualizations are a map containing the current position of observed bats, heatmaps, local convex hulls and diagrams depicting sequences of activities over time. A graphical user interface is not necessarily included by the term visualization.

1.3.4 Data Quality

The data quality undoubtedly has impact on different analyses. The availability of the z-axis may allow richer queries and descriptions of events.

It should be assessed how accurate the position data streams need to be for reliable results of the presented analyses and visualizations. Some hints should be provided on the restrictions to answer questions with regard to sampling rate, the quality of the z-axis and the spatial resolution in general. Alternative approaches to cope with data quality problems should be discussed as well.

1.3.5 Generalization

Analyzing position data streams to track animals or various kinds of objects is an important matter not only restricted to tracking bats. Hence, the system should provide a more general solution to the analysis of position data streams to be applicable in a wide range of problems.

1.4 Approach

In order to achieve all depicted objectives, biologists questions and requirements have to be examined and considered in every aspect of the presented approach and prototype. However, tailoring design decisions, presented techniques and analyses only to BATS was avoided where possible. The abstraction of techniques and notations should be particularly high to be applicable for a wide range of use cases and still as low as necessary to support the operation purpose of BATS.

1.4.1 Answering Biologists Questions

Possible answers to questions in biology, which are explained in chapter 2, can be categorized into several solution strategies.

Solving questions depending on the position of bats, e.g. whether they show territorial behavior or which places they avoid, can be supported by providing several visualizations of position data. Heat maps and local convex hulls as well as maps for the average velocity in an area are implemented in the prototype to this end. They are introduced in the fundamentals in chapter 4.

Detecting events and activities allows to examine certain aspects of hunting and social behavior. The occurence and frequency of events as well as their order can help biologists on this matter. The availability of abstract activities improves the previous categorization of *resting*, *activity* and *unknown* drastically.

Some questions can be solved by correlating conditions or rather external and internal influences of the animal to activities. Examples for external influences are weather conditions, brightness and time. Internal influences could be how saturated an animal is or whether it drank enough water. The topics supported by this correlation are for example lunaphobia, whether bats have a personality and whether they memorize previous water places as well as foraging areas in order to return to them. The techniques

to predict activities can be re-used for this purpose, since predictions also correlate influences to past behavior.

All mentioned analyses are depicted in chapter 7.

1.4.2 Live Analysis

Since the sensor network is going to collect several streams of sensor data, continuously processing the incoming streams suggests itself. The data stream and complex event processing models are suitable for such a scenario. Those models especially consider time and order, which is also useful to answer several questions of biologists. In order to provide adequate notations for different tasks, data stream processing and complex event processing are both used and need to be integrated.

After assessing several systems, the model of PIPES was chosen because its operators are well-defined and the model emphasizes time. Esper clearly had influence on the notation in chapter 5 and integration of complex event processing. The fundamentals of data stream systems are outlined in chapter 4.

1.4.3 Events and Activities

Events and activities are the basis of a big share of the presented analyses and many more to come. Since they can be used to describe the behavior of an animal, improving the detection of events and activities leads to an improved and more predictable behavior. The fundamentals are explained in chapter 5.

Events and activities have to be *detected* in a position data stream. Since a data stream processing approach was chosen, data stream operators need to be provided to interpret continuous queries as events. Notations and operators need to be supplied to aggregate events to compound events and observed activities.

Domain experts should be able to *model* events and activities. Since the presented techniques should enhance the understanding of the behavior of bats and other species, known descriptions may change over time. Therefore, the employed notations should allow easy modifications of event and activity definitions.

In order to directly exploit an enhanced understanding of the behavior of bats, activities should get *predicted* by the prototype. A more accurate knowledge of the relations between both the lengths and regions of activities leads to better predictions.

Predictions have different applications. They can support experiments by suggesting future regions to be recorded. For example, if hunting in a certain area is predicted in

advance, cameras and microphones may be moved to that area in order to record it. Moreover, it is possible to use predictions in order to activate already prepared recording equipment or to trigger experiments. Finally, techniques to predict the behavior can also help answering some of the biologists questions.

The fundamentals of predictions are explained in section 4.4 and the approach taken is elaborated in chapter 6. Chapter 8 covers the actual implementation of the detection and prediction. Since in the prototype predictions happen in a data stream processing context, their integration is outlined in section 8.5.

1.4.4 Operators

The prototype was implemented in the data stream and complex event processing context. Detecting and further processing events and activities as well as providing certain analyses to answer questions of biologists and other stakeholders requires several operators in these models.

In section 8.3 it is elaborated which operators of the data stream and complex event processing domain are necessary. Some analyses may require extended stateful operators which are explained in detail. If it is possible to translate or approximate new operators to use their behavior in other data stream systems, then such a translation should get outlined.

Since the urgency of results depends on the biologists question being asked, some analyses need real-time processing whereas others do not. For example, showing the current positions of all monitored bats emphasizes on the word *current*. On the other hand, a summary of past activities or finding home ranges does not require instantaneous answers. Section 8.3 covers real-time requirements on analyses and implementation details.

1.4.5 Simulation

Since the sensor network monitoring bats is not yet deployed, the techniques presented have to be evaluated on synthetic data.

The behavior of bats in a simulation framework was enhanced with activity-induced actions in order to move closer to actual bat behavior and to avoid clearly random movement. Parameters need to be provided to support the evaluation of the presented techniques. The changes are outlined in section 8.2.

1.4.6 Evaluation

Many techniques to model, detect and predict activities as well as to visualize and further analyze position data streams are presented in the following chapters. Moreover, one of the main objectives is to asses the impact of data quality and availability of the z-axis on different analyses. Chapter 9 evaluates the techniques and analyses with an emphasis on detection and prediction of activities as well as the impact of data quality on analyses. The evaluation is performed on synthetic data streams.

2 Scenario

The primary goal of the DFG research group 1508 (BATS) is "to gain fundamental insight into the design, construction and operation of resource-scarce, heterogenous, intelligent networks consisting of static and mobile sensors" [DFG13]. The scope of this thesis covers data analysis and data management in the context of BATS.

The following subsections give a more detailed view on the animals being tracked, explicitly mention which questions and hypotheses biologists want to investigate and outline the infrastructure which is developed to support biologists in finding the answers.

2.1 Greater Mouse-Eared Bat

The *greater mouse-eared bat* or *Myotis myotis* is one of the common bat species in central and southern Europe [RLH09]. Figure 2.1 shows one snapshot of this nocturnal mammal which prefers forests as its most important foraging habitats. The greater mouse-eared bat is a relatively large bat species. Its weight ranges from 28 to 45g and its wing-span is around 40cm [NAB, RLH09]. Myotis myotis primarily hunt flightless ground beetles in woodland. Their size and weight allows to attach around two grams of tracking equipment.

Bats and especially greater mouse-eared bats were previously examined and observed concerning activity, food spectrum, hunting and more [RLH09]. There are still many open questions which are not limited to those presented in section 2.2. The available domain knowledge helps in simulating, detecting and categorizing behavior and can also be used as a check whether known facts and measured results are consistent.

The common approach to monitor bats and other species is radiotelemetry [ADMW09], which was also used in [RLH09]. This is an improvement over previous mainly descriptive studies. However, the current approach is labor-intensive and measured positions are rather imprecise. It allows important analyses on movement and home ranges. Nonetheless, increasing precision, the amount of tagged animals, measurement periods and measurement frequency allows to further dig into the behavior.

Figure 2.1: Myotis myotis [Deu, edited]

2.2 Questions in Biology

Biologists have several assumptions on the behavior of bats and hypotheses on probable behavior. As soon as some of the statements are verified or invalidated, inevitably more questions and assumptions on behavioral patterns will arise. Hence, this section can only consist of a current snapshot of motivating questions in the hope that many more will emerge.

2.2.1 Expected Behavior to be Validated

The following behavioral patterns are assumed to be true, although they still need to be validated.

1. **Hunting**
   A bat flies in a straight trajectory avoiding trees and scrub. As soon as it hears a suspicious noise, it changes its direction to close in. It lands on the ground to grub for the suspected beetle.
   Successful Catch: The bat flies straight up and circles in the air while consuming its prey.
   Failure: The bat flies straight up and resumes hunting.

2. **Memory**
   Bats memorize previous water places and foraging areas and return to them.
3. **Avoidance of open places / Navigation**
   Bats need various obstacles to orient, hence they fly in their vicinity.
4. **Drinking**
   Bats frequent ponds in the forest to quench their thirst.

2.2.2 Probable Behavior

Some characteristics of bats are only suspected. Since the sensor network of BATS gathers a lot more position and other contextual data than the previous approach, it should now be possible to answer these questions:

1. **Lunaphobia**
   Does the behavior of bats depend on the visibility of a full moon?
2. **Personality**
   Do bats have personalities? Do conservative or wild bats exist?

2.2.3 Hypothetical Behavior

The following topics are highly speculative and to find answers should now be possible.

1. **Social behavior**
   At what age do bats start to engage in social activities?
2. **Territorial behavior**
   Do bats avoid the territory of another bat? Do they show territorial behavior to other species?
3. **Hunting School**
   Is hunting taught to a bat by its mother? Does the juvenile follow its mother to observe and learn? Is food brought to a juvenile bat until it starts to hunt for itself?

2.2.4 Further Requirements

The goal of the prototype is to support biologists in gaining more insights on animal behavior. This assistance can further be enhanced by meeting the following requirements:

1. **Experiments**
   There should be certain techniques to support experiments. For example, some events could trigger the recording with a camera or microphone.
2. **State-Of-The-Art Analyses**
   Some analyses need to be integrated into the prototype, e.g. heatmaps and local convex hulls. Moreover, adding more analyses should not take too much effort.

2.3 Setup

Since the current approach to monitor bats and other species is labor-intensive and not suitable for answering all of the stated questions of biologists, the setup of BATS is proposed. Figure 1.1 on page 2 shows the topology and architecture of the sensor network. It is comprised of three parts:

1. Mobile sensor nodes attached to the tracked animal
2. Stationary sensor nodes
3. Central station for management and data processing

The *mobile sensor nodes* are attached to bats and therefore need to be extremely light-weight to meet the restriction of two grams. The nodes also have to be energy efficient, since a weight limit narrows the battery capacity. The chosen approach is to gather positions by signal strength and phase measurements [NHK+14]. The mobile sensor nodes emit signals to be measured.

A network of ground-based *stationary sensor nodes* combines measurements to position hypotheses. Moreover, stationary sensor nodes are an interface to both mobile sensor nodes and the central station. Additional sensors can be attached in order to measure environmental data, e.g. temperature, humidity and brightness.

A *central station* is a common desktop or laptop computer communicating with the stationary sensor nodes. It stores gathered data and executes the presented methods and techniques to analyze and process position and environmental data streams.

The data exchanged between mobile sensor nodes are the ID of the tracked animal, encounters with other tracked animals (ID, time) and possibly status information (e.g. “low battery”).

The communication between mobile sensor nodes and stationary sensor nodes can be divided into two classes: Stationary sensor nodes receive the ID of a tracked animal as

well as field strength and phase to measure positions and relate them to the appropriate animal. They also exchange clock data, encounters between animals and other sensor and status data gathered on mobile nodes. Stationary sensor nodes preprocess and forward data to other stationary sensor nodes and the central station. Examples of forwarded data are animal positions as well as sensor data emitted from sensors attached to the stationary sensor nodes. Attached sensors could measure the temperature, humidity, brightness, noise as well as other environmental data.

A disadvantage of the proposed approach is that positions for animals out of range of the stationary sensor nodes can not be measured. However, radiotelemetry has the same problem. Since GPS reception is poor in forests [RPÁTSA+13] and available sensors have too much weight, changing position gathering to use GPS sensors is also not reasonable. Further miniaturization of sensors and integrated circuits certainly will allow to solve this problem in the given weight restrictions.

On the other hand, a big advantage of this setup is that it can measure positions and environmental data continuously 24/7. Compared to the prevalent approach of radiotelemetry for this purpose, it requires less manual work. Position measurements by domain experts are replaced by less time-consuming maintenance work of e.g. swapping batteries now and then. Of course, the tagging of animals ist still necessary and not automatized. As long as stationary sensor nodes are available, position measurements are possible even in forest areas without any GPS reception.

As soon as this approach is deployed and debugged in the field, an obvious optimization to work assignments also processes data on stationary and mobile sensor nodes in order to reduce the amount of exchanged data. Less data transmissions leads to a reduced energy consumption.

2.4 Summary

This chapter gave a more detailed view on the scenario in which position data streams have to be analyzed and processed. The bat species to be monitored, the greater mouse-eared bat, is introduced and biologists questions on its behavior are stated explicitly.

The questions can be categorized into several solution strategies. The first group comprises questions depending on the *position* of bats, e.g. whether they show territorial behavior or which places they avoid. Aspects of hunting and social behavior are in the group of questions solved by *detecting events and activities*. The availability of activities can also improve answers to questions in the first group. The last group

contains questions to be solved by correlating conditions or rather external and internal influences of the animal to activities, e.g. finding out whether bats have lunaphobia or whether they memorize previous water places and foraging areas.

3 Related Work

Some of the questions described in the last chapter have already been stated for different species with varying size and other types of objects, e.g. cars and pedestrians. Different limitations on size of the equipment and how to gather position data yielded different approaches to answer the questions of domain experts.

There are similarities between the presented projects and BATS on the requested analyses and how to further aggregate position data to behavior. Aspects of the implementation, for example integrating data stream and complex event processing, have also been assessed and tried before. Moreover, there are many different approaches to interpret trajectories and how to enrich them with semantic information.

3.1 Tracking Animals

Many projects tracking animals have been designed and deployed in the past. Some projects are shortly presented. Of course, there are huge differences in the approach to answer questions concerning both the size and amount of available sensors.

Livestock

The main intent of [GCP+06] is to understand the behavior of livestock in order to control it by certain stimuli. Previous attempts for cows and deer clearly showed that those animals respond to control signals, e.g. electric shocks and noises. However, they did not respond as expected, since the assumptions were not backed by knowledge on the behavior. For example, cows ran straightforward with their heads shaking or went in circles while the stimulus was applied.

In order to change that, a collar with a sensor board is attached to some of the cows in a testbed. GPS information, accelerometer and magnetometer data as well as the temperature are collected. One important analysis is the classification of the animals activities, which is also important for BATS. The classification is hierarchical and on the highest level depends on whether a cow moves. Figure 3.1 shows the classification

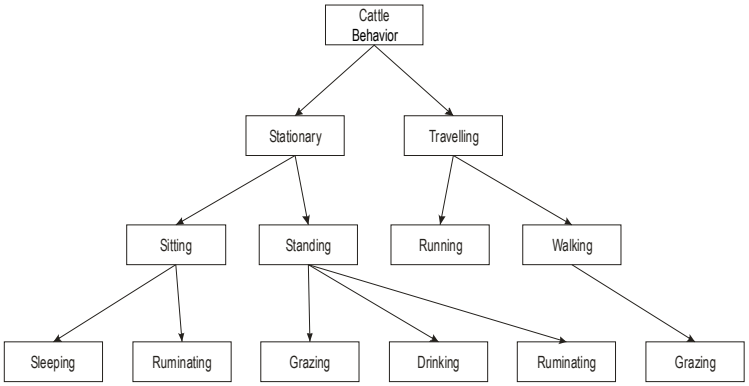


Figure 3.1: Classification of Cow Behavior [GCP+06, edited]

structure. Lower levels depend on roll angle, pitch angle and heading angle as well as changes of the angles over time. There are some important differences to BATS: Considering cows are huge, the sensor boards can have a larger size as well and can locate them by a GPS sensor. Bats are free to fly away to other areas and since their sensor nodes do not have GPS, some position data may be missing. Moreover, bats can fly and trajectories are no longer 2-dimensional.

Badgers

In [DEM+12] badgers are tracked. They are an order of magnitude smaller than cows and therefore need smaller tracking collars, in this case active RFID tags. Detection nodes are spread throughout the woods at key locations around known badger setts and latrines. Additionally, sensor nodes monitoring microclimatic conditions are deployed within badger foraging areas. The goals of this project can be categorized into two fields. On the infrastructure side, the focus is on the process to design, deploy and maintain a sensor network. The goals of zoologists involved were to gain knowledge in movement patterns, social interactions and the correlation of badger activity with night-length. Movement patterns in [DEM+12] are sightings of badgers at sensor nodes during the day. Sightings were combined to temporal density plots for sensor nodes near setts, latrines and altogether. Social interactions in different situations, e.g. for setts and latrines, are recognized in a weighted social graph. The weight represents the time of contemporaneously being spotted at detection nodes. Communities are detected with an algorithm presented in [BGLL08].

The setup is similar to BATS. Sensor nodes are directly attached to the animal and positions are measured by stationary sensor nodes. They split the stationary sensor

nodes of BATS into detection nodes and sensor nodes gathering environmental data. Instead of a central station they provide a 3G link and use biologists vehicles as mobile sinks.

Optimizations are performed in stages taking turns on the hardware and software side. They present and evaluate adaptive sensing in simulations and actual field tests. They also demonstrate the impact of hardware improvements on software optimizations. The evolution of the project led to cost reductions from 372.50 USD to 10.30 USD on average to monitor one badger for four weeks. The costs to get to the woods tagging animals is higher than the maintenance costs.

[DEM+12] concludes that it is important to continually collaborate with domain experts. Rapid prototyping and gradually improving hardware and software proved to be a successful approach to build their sensor network. They advise to pay attention to network maintenance costs from the beginning. The emphasis of the paper lies in the sensor network. However, the zoologists goals and presented analyses are also good examples for analyses on position data streams.

Rats

In [TOBW08, OTBW08, LTM06, BLBGW10] rats are monitored. Rats live in underground burrows, hence radio propagation is rather limited. The setup contains sensor nodes attached to the animals and base stations to collect data. Base stations are placed at the exits of the rat burrow. As soon as a rat passes one, the measured data on the mobile node is transmitted to the base station. Mobile nodes ("Ratpacks") need to be small and light-weight to fit on a rat and not restrict the animal in its natural movements.

Biologists wanted to investigate several phenomena. They wanted to gain insight into social interactions of rats, thus answering the questions when, where and how long rats meet. Acceleration sensors described the motion and social behavior of a rat. They also added microphones to examine vocalizations.

There are similarities to BATS. Both projects focus on size, weight and battery usage of mobile nodes in order to fit on the animal. It is not assumed that animals always connect to base stations or stationary nodes. Encounters of animals are detected and data is exchanged between mobile nodes.

3.2 Detecting Events and Activities

The approach to detect events and activities outlined in chapter 5 is similar to existing techniques.

3.2.1 MavEStream

In [JAC07, CJ09] data stream processing and complex event processing are integrated. Data stream systems and complex event processing systems both operate on streams, work in-memory and emphasise on temporal locality either with windows or consumption modes. Their biggest differences lie in the semantics of the models. MavEStream cascades the output of regular data stream processing interpreted as an event into an event processing engine. Their approach is outlined in figure 3.2.

Stage 1 in this model corresponds to a general purpose data stream management system. Afterwards, so called *computed events* are generated in Stage 2. Those computed events are defined by regular continuous queries in the data stream processing model, but are interpreted as primitive events. Hence, Stage 2 serves as a bridge between data stream and complex event processing. Stage 3 corresponds to a regular complex event processing. Rule processing, which automatically triggers sequences of actions for specific alarm types (events) and defined conditions, is usually part of the CEP system and referred to as Stage 4.

If an application domain benefits from both models, which undoubtedly is the case with BATS, then the presented approach allows to use a proper notation for each abstraction level or stage.

Nonetheless, some integration issues remain. The *event generation* out of data streams allows huge amounts of computed events, which cannot be anticipated by CEP engines [JAC07]. Moreover, the timestamp of computed events are an important implementation decision, since one computed event may be derived by a large group of data tuples. Integrating both models may result in separate programs which need to communicate. Therefore, *address space issues* emerge. Stage 2 is either a separate middleware or added to the address space of one of both systems to be integrated. In any case, common data structures and Inter Process Communication (IPC) or Remote Procedure Calls (RPC) have to be taken into account. The obvious solution to this problem combines both systems and the middleware in one address space. However, this depends on the availability of source code or object files as well as licensing.

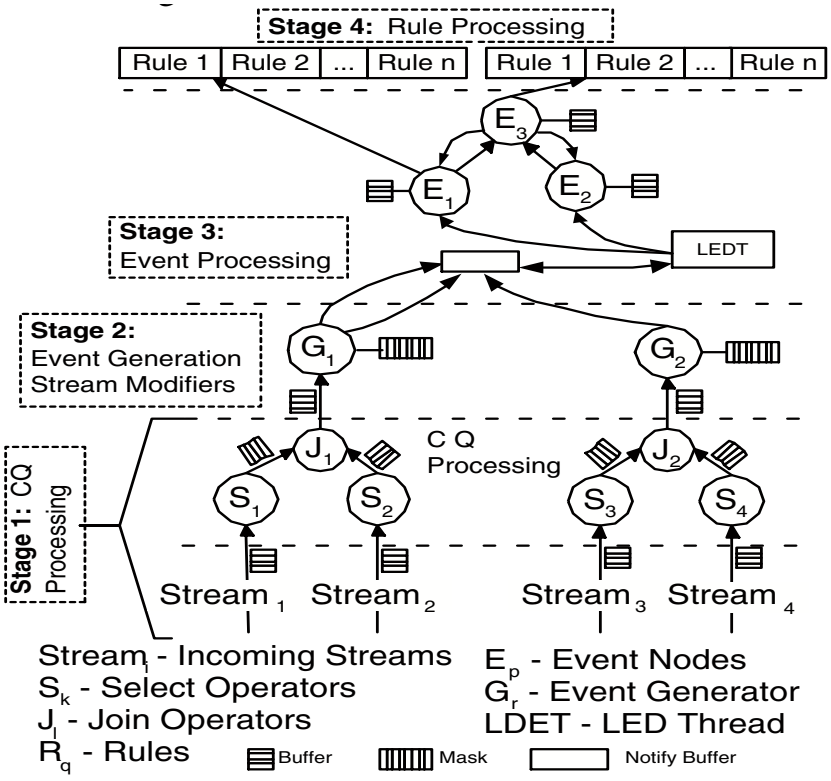


Figure 3.2: Integration Model of MavEStream [JAC07]

There are huge similarities between MavEStream and the detection of events in the presented prototype. Computed events correspond to low level events and regular complex event processing corresponds to the creation of high level events. In the prototype, events are further aggregated to activities and several extensions are added to simplify the usage of AI methods. However, MavEStream may as well be used as a backend prior to activity recognition. The primary goal of this thesis is to analyze and process position data streams and not integrating data stream and complex event processing.

3.2.2 Deriving Spatio-Temporal Query Results in Sensor Networks

In [BBBB10] it is determined how to derive spatio-temporal query results in sensor networks. This appraoch was proposed to be used in moving object databases without explicitly mentioning data stream systems. However, its different parts are analogical to low level and high level events restricted purely to position data streams.

Spatio-temporal predicates, e.g. *inside(pos, **R**)* or *meet(pos, **R**)*, correspond to low level events. They take the uncertainty of the actual position of an object into account by returning either *true*, *false* or *maybe*. A satisfied predicate is guaranteed to be fulfilled, whereas the unsatisfied predicate is guaranteed not to be fulfilled. *Maybe* is returned if no clear deduction is possible. A region R also is divided into different disjoint subsets:

- The interior R^I covers all points within the region.
- The border R^B contains all points of the line that delimits the region.
- The exterior R^O contains all points which are not in R^I or R^B.

Points do not have a border. Predicates can use for example the intersections of the interior, border and exterior between a point and a region to describe their topological relationship and the predicates *disjoint*, *meet* and *inside*.

Predicates can be combined as a *concatenation*. The sequence of spatio-temporal predicates is called a *spatio-temporal development*. They can be interpreted both as a high level event, since the concatenation is a simple pattern, and also as an activity.

The notation and vocabulary of [BBBB10] will be used in chapter 5 to describe certain aspects of high and low level events.

3.3 Tracking Objects

Detecting and tracking objects in traffic is prevalent in driver assistance systems to provide intelligent cruise control or to warn of a crash or even prevent one. These systems typically are proprietary and rarely share the same sensors. Deployed in a car they might even detect the environment independently and redundantly. [Bol11] presents a framework for the development of advanced driver assistance systems to avoid these problems.

The framework applies a data stream processing approach. The data model is based on PIPES (section 4.2), which is also used in this thesis. It has been extended to be bitemporal. This is necessary, since for example for a prediction the time of the creation of a prediction as well as the time of the predicted occurence of its content are both crucial for the interpretation. The time of the creation of the prediction corresponds to the validity of the prediction since it can overwrite old revisions or be invalidated by newer forecasts. The predicted time is at least as important as the creation time since actual measures to be taken depend on it.

The most important similarity to BATS is adding and updating a state of each tracked object. In BATS this is important for the detection of activities. This framework detects objects and adds attributes, e.g. velocity and vicinity. Those attributes are stored in a context model. Moreover, cycles are allowed by adding a *broker operator*. They are defined procedural rather than descriptive to improve modeling and understanding. In BATS cycles focus on activity detection.

In summary, BATS and [Bol11] both base the data model on PIPES and explicitly allow state of objects and cycles in queries.

3.4 Semantic Trajectories

Different approaches to model and analyze semantic trajectories have been surveyed in [PSR+13]. The paper introduces the concepts and definitions surrounding semantic trajectories and thereby a common vocabulary for depicted techniques. This thesis also adheres to the vocabulary where possible.

"A *raw trajectory* is a trajectory extracted from a raw movement track and containing only raw data for its Begin-End interval." [PSR+13] Sometimes, raw data of a movement track is missing, which for example is the case of hardware or software malfunction. Such an occurrence is called a *hole*. If this happens for just a short period, then a hole may be filled, e.g. by interpolation. Furthermore, data may be missing on purpose, for example if a user switches off his GPS-enabled device or a sensor node on a bat is in an energy-saving mode. A period of data missing on purpose is called a *semantic gap*.

Adding additional knowledge to a trajectory is called *semantic enrichment*. An external source of knowledge provides contextual data, e.g. a database of terrain features, and is therefore called *contextual data repository*. Additional knowledge added to a trajectory is called an *annotation*.

Bearing all previous definitions in mind, "a *semantic trajectory* is a trajectory that has been enhanced with annotations and/or one or several complimentary segmentations" [PSR+13].

Figure 3.3 is an example to apply all definitions. The journey starts and ends at Hotel Zola in Paris. The raw trajectory contains a list of all measured points at its time instants. It is divided into several segments which were divided by periods of slow or no movement, e.g. in order to visit points of interest. A database of tourist features acted as a contextual data repository which linked GPS data from periods of slow/no movement to points of interest, e.g. the first stop at the Eiffel Tower. Segments of sightseeing stops

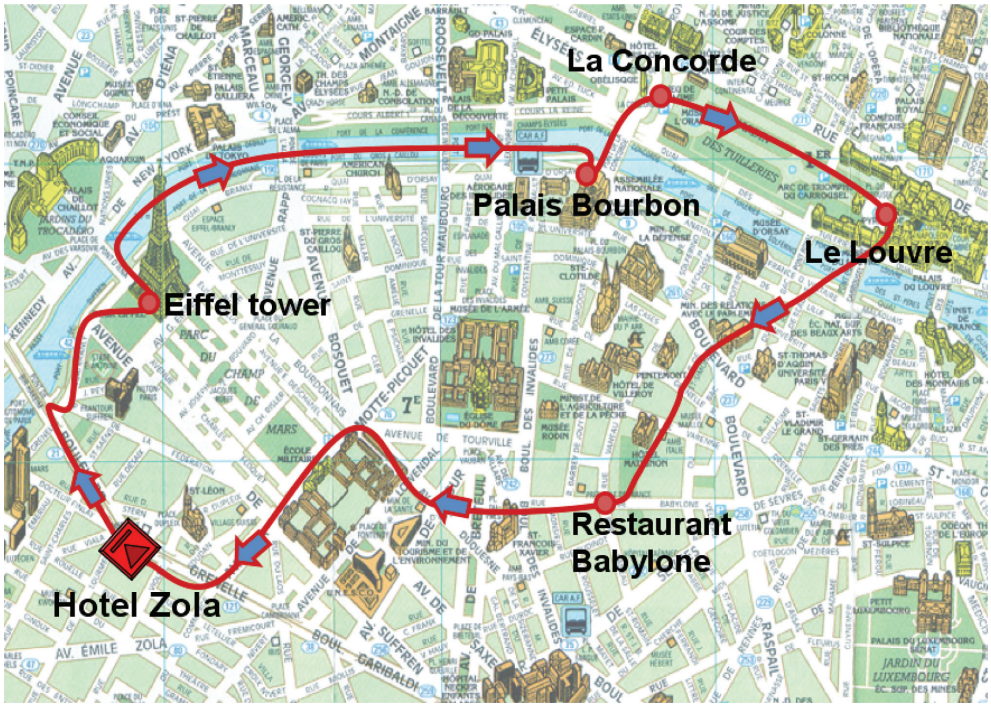


Figure 3.3: 2D Visualization of a One-Day Spatial Trace Left by a Tourist Visiting Paris [PSR+13]

were annotated with the corresponding sights. The whole annotated and segmented sightseeing tour is the resulting semantic trajectory.

Semantic trajectories in the following chapters are represented by events. Segments likewise are events, too. However, in order to match the definition of hard segmentation without overlaps in [PSR+13], the corresponding events have to be explicitly defined for this purpose. Annotations for semantic trajectories are equal to annotations for events. A database of user-defined regions, e.g. watering places or areas with special terrain features, streams of sensor data as well as implicit contextual data in event and query definitions act as a contextual data repository. Of course, the prototype may be extended to other data sources.

Several techniques in [PSR+13], e.g. trajectory map-matching, are currently neither implemented in nor relevant to the prototype. Since requirements change over time, the survey may prove to be a helpful source of information to cope with a new situation in the future.

3.5 Summary

This chapter introduced several approaches to animal tracking. The size of observed animals and thereby the possible and deployed size of attached sensor nodes varied considerably. Moreover, movement patterns ranged from rats living in underground burrows to fenced cows as well as to badgers which were allowed to move without restraint. Different requirements yielded different solutions which were tested in practice. The applicability of aspects and solutions to BATS were outlined for each project.

The approach of [Bol11] to track objects in driver assistance systems shares similarities to the prototype described in the following chapters. The data model is based on PIPES. However, the most important similarity to BATS is adding and update the state of each tracked object. The broker operator supports cycles in data stream processing.

Detecting and further processing events in a data stream processing model requires to integrate complex event processing. The approach of MavEStream was presented and compared to the solution in the prototype. Since events and activities are mostly aggregated out of position data streams, some of them resemble spatio-temporal predicates. [BBBB10] determines how to derive spatio-temporal query results in sensor networks. The notation is adopted for descriptions.

Finally, definitions and concepts in the survey [PSR+13] on semantic trajectories were outlined. The vocabulary as well as surveyed ideas are incorporated into the following chapters and the prototype.

4 Fundamentals

This chapter introduces the fundamentals of the solutions elaborated in the following chapters. First of all, data stream systems and examples for both prevalent flavors are presented. Important characteristics of the examples PIPES for the data processing model and Esper for complex event processing are highlighted, since they had severe impact on the prototype. They are followed by explanations of several techniques of machine learning in order to allow predictions of behavior. Finally, visualization techniques requested by biologists are described.

4.1 Data Stream Systems

A data stream system (DSS) consists of a data stream management system (DSMS), data stream definitions and stored continuous queries. Continuous queries provoke data stream systems to process incoming data immediately in order to produce an output stream. They work on transient data and relational data stream systems interpret streams as a sequence of tuples. A DSMS manages schema definitions of incoming streams as well as continuous queries and offers access protection. Similar to DBMS, DSMS are typically built paying attention to application independence. [MW13]

Currently, there are two main flavors in this area: Data stream processing and complex event processing (CEP). These two models have a lot in common on how they work. Differences lie mostly in their origins, vocabulary and main use cases.

Data stream systems can be considered as an evolution to traditional database systems. They process streams of different sources to new data streams as output [CM12]. The query languages mostly are designed to resemble SQL and many transformations work alike. On the other hand, complex event processing regards incoming data as events happening in the external world. These events have to be filtered and combined in order to find underlying higher-level events [CM12]. Hence, pattern matching of events is the main goal which surfaces in the supplied APIs and query languages.

Since some use cases – including BATS – can benefit from both models, there have been attempts to integrate them. One example of this integration is MavEStream, which was explained in section 3.2.1.

Apart from different names, many operators of the relational world are used in DSS as well. Some non-blocking operators in DBS are already implemented in a stream-oriented way to speed up processing. For example, projection or selection can be applied during the retrieval of a relation. Blocking operators, for example aggregation and join, have to be revised.

Continuous queries require operators to emphasize on time and order. For the data stream processing context, the *Window* operator has been added to fulfill this requirement. It captures temporal locality, either time-based or count-based. For example, a window allows to return the velocity of an object by using two subsequent positions in a stream.

By using windows, blocking operators are once again possible. For each time instant, only a finite amount of tuples has to be processed. Of course, this leads to redefined semantics of blocking operators. For example, aggregations are processed according to window definitions for each time instant instead of one final value as a result. The output of a join is returned regularly and only infinite windows and infinite main memory allow to fully join two infinite streams.

[CM12] examines data stream processing and complex event processing systems in detail with respect to functional and processing models, deployment as well as offered operators.

4.2 PIPES

One important example of a data stream system is PIPES [Krä07a, Krä07b]. The model was chosen for the prototype, since its operators are well-defined and the model emphasizes on application time. The stream representations enable arrival of elements at the same time instant. Even concurrent identical elements are anticipated.

4.2.1 Stream Representations

In [Krä07a] several stream representations are introduced in order to use for each situation the most convenient definition. *Raw streams* are input streams registered to the system. Internal representations are divided into *physical* and *logical streams*. Transformations

from raw to logical and physical streams as well as physical to logical streams are provided to ensure semantic equivalence of the definitions.

Raw streams are potentially infinite sequences of elements (e, t), whereas e is a tuple and t an associated timestamp. They are non-decreasingly ordered by timestamps. Raw streams are divided into *base streams*, i.e. streams which act as an input to logical and physical plans, as well as *derived streams* which are output streams of logical and physical operators [Krä07a, MW13].

Logical streams are order-agnostic multiset representations of raw or physical streams and emphasize on the validity of tuples at time-instant level [Krä07a]. They are a potentially infinite multiset of elements (e, t, n) whereas e is a tuple, t is an associated timestamp and n is the multiplicity of the tuple.

Physical streams are a more compact representation of their logical counterpart. They are a potentially infinite multiset of elements $(e, [t_S, t_E))$, whereas e is a tuple and $[t_S, t_E)$ is its time interval.

4.2.2 Snapshot-Reducibility

In order to apply algebraic query optimizations already known from relational operators, data stream operators can be analyzed in regard to whether they resemble the behavior of a relational, non-temporal counterpart for each time instant [Krä07a].

A snapshot of a logical stream S^l at a time instant t can be considered an instantanious relation which represents the bag of all tuples of S^l valid at the time instant t. The timeslice-operator τ computes the snapshot of S^l [Krä07a].

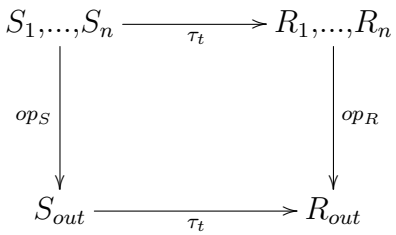


Figure 4.1: Snapshot-Reducibility [Krä07a]

Figure 4.1 sketches the definition of snapshot-reducibility. A stream-to-stream operator op_S with the inputs S_1,...,S_n is snapshot-reducible, *iff* at any time instant t the snapshot of the results of op_S is equal to the results of applying its relational counterpart op_R to the snapshots of the inputs S_1,...,S_n at the time instant t [Krä07a].

4.2.3 Data Stream Operators

PIPES provides many operators to use in continuous queries. *Filter* corresponds to its relational counterpart *selection*. *Map* covers the relational *projection* as well as replacing or creating attributes which values may depend on the content of the mapped tuple. Mapping does not affect the timestamp.

The semantics of *cartesian products* and thereby joins are different from the relational world. A cartesian product combines elements of both input streams whose tuples are valid *at the same time instant*.

PIPES provides time-based sliding windows as well as count-based sliding windows. Furthermore, count-based sliding windows can also be partitioned.

4.3 Esper

Esper [Espa] is a great example for the second flavor of data stream systems, complex event processing. It can be considered the leading open-source CEP provider [CM12]. Esper is available as a standalone application. Libraries are also available to embed it in a Java environment (Esper) or .NET (NEsper) [Espa].

The Event Processing Language (EPL) is a rich declarative language for rule specification which is part of Esper [CM12]. It provides *patterns* for logical and temporal event correlation, among others conjunction, disjunction, sequence and negation. An important operator in EPL is *every*. A pattern without it stops looking for matches as soon as one is detected. *Every* enforces to keep looking for matches. Furthermore, Esper provides *event consumption*. If an event was consumed, it is no longer available for other patterns to match.

Esper supports a subscription-based delivery to listeners (push) as well as a receive-based delivery using iterators (pull) [Espa].

Some of the notation in chapter 5 is borrowed from EPL.

4.4 Techniques in Machine Learning

There are several approaches in the field of machine learning which can also be applied in the present scenario. Hidden Markov models are introduced in order to present similarities between problems in BATS and solved problems in HMMs. Bayesian networks and Dynamic Bayesian networks are the basis of behavioral predictions presented in the

following chapters. Since the Kalman filter can improve the accuracy of measured position data, it is shortly introduced as well.

4.4.1 Hidden Markov Models

A Hidden Markov Model is a doubly stochastic process that models both actual states and observable emissions. The states, as the name suggests, are hidden. Moreover, transitions between states are a regular Markov model. The emissions are observed with a certain probability depending on the actual state. Hence, the actual states can only be observed through the emissions or rather through another set of stochastic processes [RJ86].

This short description of a Hidden Markov Model matches the situation of detecting certain activities or states of e.g. a bat by looking at some available sensor data. It may not be possible to observe the actual thought process and current state of an animal. But the results of this thought process, the actual behavior, *is* observable. Furthermore, the thought process most probably not only depends on the last time instant and last state. Even if it does, the modeled behavior might be dissimilar to reality and not reproduce this dependency. Hence, giving and updating certain probabilities to observable emissions and transitions between modeled hidden states allows to approximate the unknown actual model. Hidden Markov models are applied in many uses cases, for example in speech recognition, GSM and UMTS.

4.4.1.1 Elements of a Hidden Markov Model

Each model contains a finite set of *states* Q. Each state possesses some measurable, distinctive properties [RJ86]. The state is represented by one discrete random variable. If there is more than one (discrete) random variable, e.g. parallel processes, then some *mega variable*, a cross product consisting of all of them, has to be created [RN04].

States are connected by *transitions*. For each time instant, a new state is entered based upon a transition probability distribution A only depending on the previous state [RJ86].

For each time instant, there is one current state and one observation output, *emission*, is produced according to a probability distribution B only depending on the current state [RJ86]. The set of possible emissions V is one discrete random variable. If in reality it is more than one (discrete) random variable, then one mega variable has to be created. For

example, the state of the weather may have different outcomes for visibility of the moon and whether it is raining.

Finally, there has to be an initial state distribution π. A Hidden Markov Model λ can now be described by $\lambda = (Q, V, A, B, \pi)$.

4.4.1.2 The three problems of Hidden Markov Models

An HMM is not created for its own sake, but to solve a problem at hand. Three key problems of interest in real world applications have been identified [RJ86]:

1. Given an observation sequence $O = (O_1, O_2, \ldots, O_r)$ and the HMM λ, what is the probability of the observation sequence $P(O)$?
   In the context of BATS, this problem is equivalent to finding highly improbable observation sequences which deserve a closer look.
2. Given an observation sequence $O = (O_1, O_2, \ldots, O_r)$, what is the optimal sequence or rather *path* of states that is produced by the given observation sequence?
   Applied to BATS, this problem is equivalent to identifying behavioral states from the available sensor data.
3. How should the model parameters A, B, π in λ be adjusted to maximize $P(O)$?
   This problem is equivalent to the main cause of using AI methods in BATS: predictions. By assuming that past behavior approximates future behavior, the λ with the highest $P(O)$ for a previous observation sequence $O = (O_1, O_2, \ldots, O_r)$ can be used to predict future $O_{r+1}, O_{r+2}, \ldots$ and states.

All of these key problems already have been solved. Finding the probability of a given observation sequence can be done by using the forward-backward procedure. This algorithm takes N^2T steps, where N is the number of states $N = |Q|$ and T is the length of the observation $T = |O|$ [RJ86]. In order to find the most likely path of states producing a given observation sequence, it is possible to apply the Viterbi algorithm [FJ73, RJ86]. For adjusting model parameters, the Baum-Welch method is a well-known iterative approach [RJ86].

4.4.1.3 Applicability for BATS

The set of all possible states and all possible emissions has to be represented by one discrete random variable. A lot of different random variables go into the actual state of a forest and decision processes of a bat. A small subset of variables could be: time of the

day, weather, visibility of the moon, availability of food, saturation and thirstiness of a bat, fatigue, natural enemies and more. Some of these random variables are interdependent. The random variable for observations also has to include a lot of sensor data in order to detect as much of a state as possible.

Since the mega variables are cross products of all of their random variables, this results in huge numbers of states and emissions and enormously big tables for probability distributions for transitions and observations. A more applicable solution is presented in section 4.4.2.

Even though the second of the three problems of hidden Markov models is solved, its solution is not applicable for identifying activities in BATS. The complexity of the viterbi algorithm is $\mathcal{O}(T \cdot |N|^2)$, whereas T is the length of the observation and $N = |Q|$ is the number of states *including* emissions. N is highly dependent on the number of possible positions and thereby on the size of the observed area as well as the precision of measurements. Hence, the solution does not scale well with the observation area. The next chapter emphasizes on the notation of several abstraction levels to detect events and activities. Events can be considered to be an aggregated view of observations. Nonetheless, the implementation of activity detection takes advantage of the proposed notation of activities.

4.4.2 Bayesian Networks

Bayesian networks exploit the fact that some random variables in a model often are independent or slightly dependent on other random variables. They are based on Bayes' theorem: [HKP11]

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

A Bayesian network is a directed acyclic graph in which every node is labeled with information on its probability distribution. It consists of the following elements: [RN04]

1. Nodes: A set of discrete or continuous random variables.
2. Edges: A set of directed edges connecting two nodes $X \to Y$.
   Y is dependent on X, X is a parent node of Y.
3. For every node X_i there is a conditional probability distribution $P(X_i|parents(X_i))$.

A *conditional probability table* (CPT) represents a conditional probability distribution of discrete random variables. If in a CPT all used random variables are measurable, the CPT can be generated automatically.

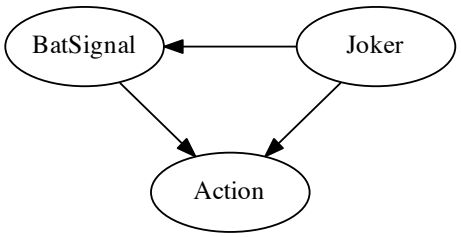
Typically, modeling of Bayesian networks is done by educated guesses on dependencies. [RP87, CH92] present approaches to automatically build the structure of Bayesian networks from statistical data.

4.4.2.1 Example

Figure 4.2 shows a basic Bayesian network with CPTs for all random variables. The Bayesian network represents a scenario in which a delinquent (Joker) frequently commits crimes. An alarm system (BatSignal) has been deployed to quickly mobilize massive police forces and special operations (Action). The alarm sometimes fails. Since special operations focus on the delinquent, they are more successful in mobilizing forces if he is operating.

Of course, the Bayesian network can be used to calculate probabilities for each random variable:

$$
\begin{aligned}
P(J) &= 0.3 \\
P(S) &= P(S|J)P(J) + P(S|\neg J)P(\neg J) = 0.34 \\
P(A) &= P(A|J \wedge S)P(J \wedge S) + P(A|J \wedge \neg S)P(J \wedge \neg S) \\
&\quad + P(A|\neg J \wedge S)P(\neg J \wedge S) + P(A|\neg J \wedge \neg S)P(\neg J \wedge \neg S) \\
&= P(A|J \wedge S)P(S|J)P(J) + P(A|J \wedge \neg S)P(\neg S|J)P(J) \\
&\quad + P(A|\neg J \wedge S)P(S|\neg J)P(\neg J) + P(A|\neg J \wedge \neg S)P(\neg S|\neg J)P(\neg J) \\
&= 0.35
\end{aligned}
$$



Joker	BatSignal S T	F
F	0.1	0.9
T	0.9	0.1

Joker J T	F
0.3	0.7

Joker	BatSignal	Action A T	F
F	F	0.05	0.95
F	T	0.8	0.2
T	F	0.2	0.8
T	T	0.95	0.05

Figure 4.2: Bayesian Network: Fighting Crime in Gotham City

It can also be used the other way around, e.g. finding the probability of the delinquent operating dependent on whether the alarm went off:

$$P(J|S) = \frac{P(S|J)P(J)}{P(S)} = \frac{P(S|J)P(J)}{P(S|J)P(J) + P(S|\neg J)P(\neg J)} = 0.34$$

4.4.2.2 Dynamic Bayesian Networks

Bayesian networks describe conditional dependencies between random variables. *Dynamic Bayesian Networks* (DBN) extend Bayesian networks with time and allow more compact representations of states and CPTs than hidden Markov models. A DBN consists of three types of probability distributions: [RN04]

1. Unconditional (initial) probability distribution of state variables X: $P(X_0)$
2. Conditional probability distribution for transitions between states: $P(X_{t+1}|X_t)$
3. Sensor Model for Emissions E: $P(E_t|X_t)$

Hidden Markov models are a special case of a Dynamic Bayesian network. All DBNs can be transformed into HMMs [RN04]. Chapter 6 uses Dynamic Bayesian networks to model predictions of activities.

4.4.3 Kalman Filter

The Kalman filter [Kal60] allows to improve measured data over time. Instead of emitting single measurements, the Kalman filter uses a series over several (imprecise) state vectors to generate an estimation of the actual state vector [Bol11]. The state vectors contain continuous random variables, for example positions or velocity.

A state transition model as well as a sensor model for measurements has to be supplied. The state transition model covers the underlying physical process to be observed and the sensor model contains assumptions on noise [RN04]. Hence, the Kalman filter can be viewed as a Dynamic Bayesian network with continuous state transitions.

The Kalman filter may be useful for BATS in order to improve measured position data. Important state variables are the position X_t, the velocity $\dot{X}_t$ and the measured position Z_t. Figure 4.3 depicts a state transition and sensor model for this scenario. An estimated position for the time instant $(t + \Delta)$ using the actual position and velocity at time t can be written as the following equation: [RN04]

$$X_{t+\Delta} = X_t + \dot{X}_t\Delta$$

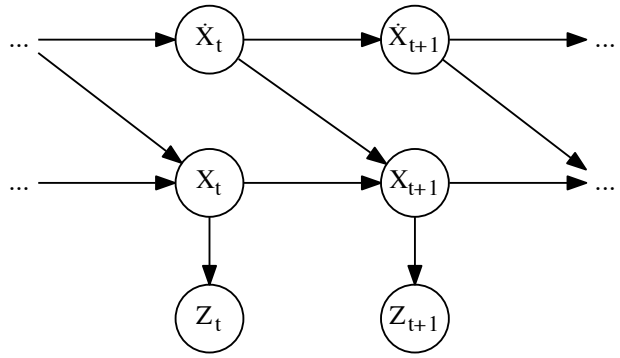


Figure 4.3: State Transition and Sensor Model of Position Data [RN04]

The Kalman filter uses this model to find $\dot{X}_t$ and X_t which resemble the measured values Z_t most closely. Its usage in BATS should be evaluated on real data measured in the field.

4.5 Visualization

There are several state-of-the-art methods to visualize position data. This section introduces heatmaps as well as local convex hulls. Both analyses are provided in the prototype.

4.5.1 Heatmaps

A simple approach to visualize a table of data is to show it all at once [Yau11]. There are several types of heatmaps. A *heat matrix* is a typical grid displayed in colors instead of values. Figure 4.4 shows an example of a heat matrix. Each cell covers a rectangular segment in the whole observed area. Measured positions are related to their respective cell. The number of measured positions in a cell is the value to be visualized.

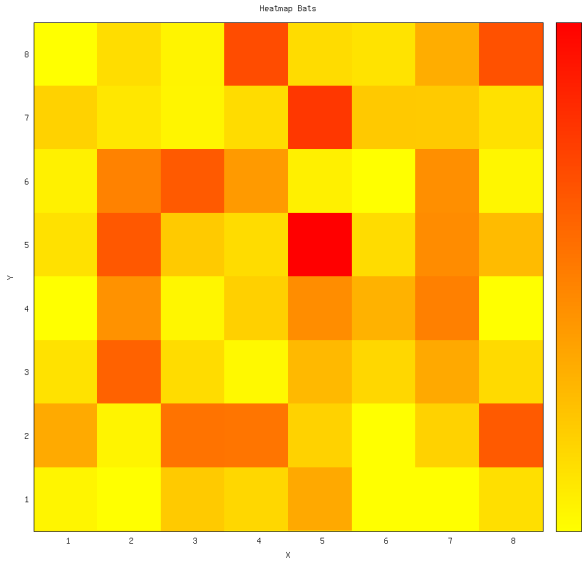


Figure 4.4: Sample Heatmap Generated by the Prototype

Another common approach are *choropleth maps* [Yau11]. Instead of rectangular cells, regions of arbitrary shape are colored. Their typical application is to visualize values by country on a world map as well as by state, county or other entities. In BATS, the regions may be user-defined areas in a forest. Figure 4.5 is an example of a choropleth map. It shows how many German students do a semester abroad per host country.

Interesting points or clusters easily attract the viewers attention in both options.

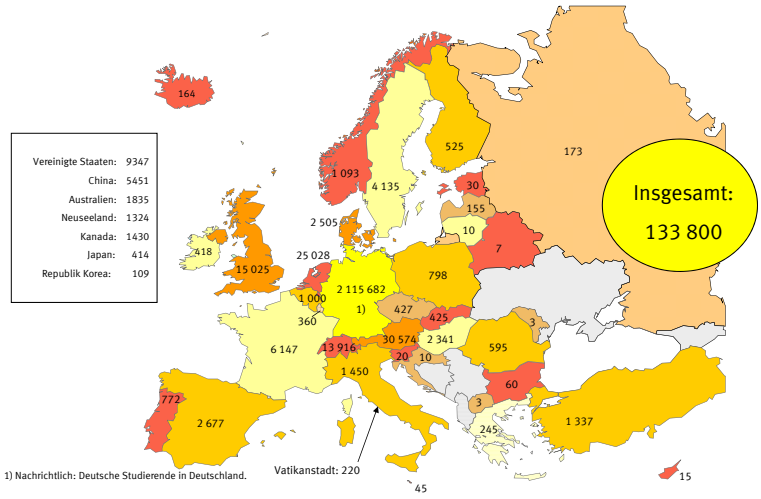


Figure 4.5: Choropleth of German Students Abroad per Country [Sta13]

4.5.2 Local Convex Hulls (LoCoH)

Estimating home ranges and utilization distributions is an important analysis of position data in biology. A home range can be defined as "the normal area used by an animal in its life activities" [ADMW09]. There are several state-of-the-art techniques to tackle this task, mainly standard kernels, Minimum Convex Polygons (MCPs) and *Local Convex Hulls (LoCoH)*.

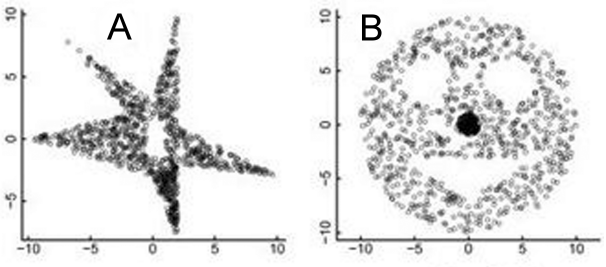
Usually, an MCP has to cover 95% of all data points to define a home range [GFRC+07]. MCPs are widely employed despite providing a poor fit to data when the home range of an animal or the distribution of a population is strongly non-convex [GW04].

Local convex hulls generalize the MCP method. They are able to identify hard boundaries, e.g. rivers or cliffs, and converge to the true distribution as the sample size increases [GFRC+07]. Unlike MCP, they can display holes in the distribution, e.g. an area within a home range which is unhabitable or avoided for other reasons. LoCoH is a family of algorithms [GFRC+07, GFR13]:

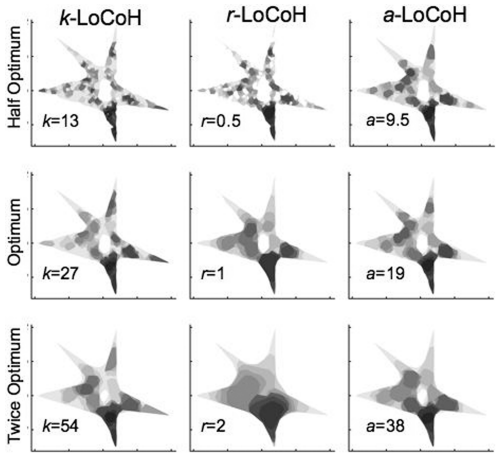
1. **Fixed k-LoCoH** ("fixed-number-of-points") creates hulls from the $(k-1)$ nearest neighbors to the root point. It is also known by its previous name k-NNCH introduced in [GW04].
2. **Fixed r-LoCoH** ("fixed-sphere-of-influence") uses all points within a fixed radius r of each reference point.

3. **Adaptive a-LoCoH** ("adaptive sphere-of-influence") creates hulls from all points within a radius of a such that the sum of the distances of all points is less or equal to the radius a.

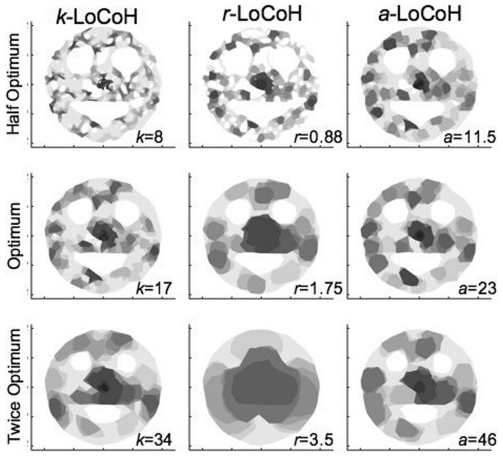
Figure 4.6 illustrates the results of the mentioned LoCoH methods. Two datasets are analyzed with k-LoCoH, r-LoCoH and a-LoCoH in several configurations. Clearly, boundaries as well as holes are detected in most example configurations. Nonetheless, which of the methods to choose as well as a proper parameter has to be decided by domain experts.



(a) Sample Data [GFRC+07]



(b) Illustrations of Utilization Distributions Constructed of Data Set A [GFRC+07]



(c) Illustrations of Utilization Distributions Constructed of Data Set B [GFRC+07]

Figure 4.6: Sample Data and Illustrations of Utilization Distributions Using LoCoH

An R script for the three LoCoH methods is available at [GFR13]. A newer version of LoCoH incorporating time - T-LoCoH - is presented in [LTG13].

4.6 Summary

This chapter presented the fundamentals of solutions which are elaborated in the following chapters. Data stream systems are introduced, since the prototype adheres to this approach. PIPES is an example for the data stream processing model. Its data stream operators are well-defined and it emphasizes on application time. Esper is an example for the second flavor of data stream systems, complex event processing. EPL is part of Esper and provides patterns for logical and temporal event correlation.

Some techniques in the field of machine learning were presented. Hidden Markov models and Dynamic Bayesian networks are introduced in order to provide predictions of behavior. Using Kalman filters is suggested in order to increase the precision of measured position data.

There are several state-of-the-art methods to visualize position data. Regular heatmaps as well as choropleth maps support highlighting several characteristics of the observed area, for example utilization or velocity. Local convex hulls are used in order to estimate home ranges and utilization distributions. LoCoH is a family of algorithms. All three algorithms are shortly explained. A newer version incorporating time is referenced.

5 Events and Activities

Animal behavior is characterized by certain events and activities. Their detection allows biologists to test hypotheses on animal behavior and to draw new conclusions.

An *event* is a message that something happened at a specific point of time or timespan. If it belongs to a tracked object, it can and should be attributed to it. However, events not belonging to any object do not need this correlation. Several events attributed to the same object are allowed to happen at the same time. Events do not need to happen at every time instant.

This definition of an event allows that atomic events, e.g. entering a region, as well as behavior lasting for some time, e.g. making a pause, are covered. Since some events for weather or generally external influences can rarely be attributed to a specific animal, this correlation is optional and depends on the modeled event. Several simultaneous events for the same object are allowed, since event definitions may cover different aspects of behavior.

An *activity* is more restricted. Every object performs some activity at each time instant. Two activities cannot be valid for the same object at the same time. A notification of an activity has to include the related tracked object. Hence, activities allow to model more abstract, mutually exclusive descriptions of the behavior of an object. A bat can not hunt and sleep at the same time.

5.1 Motivation

Figure 5.1 shows a possible categorization of bat behavior. It was elaborated in discussions with biologists. Previous detection techniques allowed to group behavior into the activities *resting*, *activity* and *unknown* [RLH09]. Being able to model and detect more complex behavior allows to further examine it, relate actual behavior to external influences, e.g. environmental conditions, and to predict it more reliably.

Parallelograms in figure 5.1 are supercategories and correspond to previous detections. Ellipses represent additional activities one level below. Unfolding behavior even more yields to events which may or may not be interpreted as an activity. For example, in

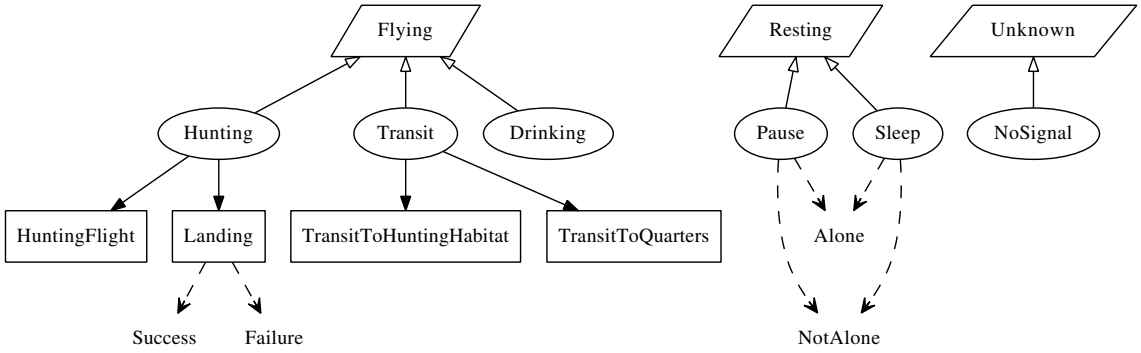


Figure 5.1: Categorized Bat Behavior

order to hunt, a bat flies through the woods until it hears a suspicious noise. This flight is interesting of its own, but it is often interrupted by successful or failed catching attempts.

Modeling hunting flights and landings as events allows to still view the whole process as hunting and to examine short timespans of behavior separately. If a set of events can be considered to be a more detailed explanation of an activity, then they can be modeled as subactivities or more generally nested activities. However, this is only reasonable if all events to be modeled as subactivities really comply with the definition of an activity, especially mutual exclusion. Otherwise, events can help to describe the characteristics of an activity, e.g. hunting consists of flights at a certain speed level interrupted by successful and failed attempts to catch an insect.

Therefore, also sample events drawn as boxes were added to figure 5.1. Additional semantic information should be annotated to events and activities whenever detectable. Examples are frameless nodes in the figure, e.g. whether other bats rested at the same spot or whether a catching attempt of an insect was successful. The events $TransitToHuntingHabitat$ and $TransitToQuarters$ are a more detailed explanation of an activity. They do not cover all types or segments of transit flights and are therefore not modeled as subactivities. Nonetheless, they allow to semantically enrich the transit activity and support the detection of a transit flight.

Table 5.1 shows a selection of events to be detected in the prototype. Events are defined if their detection helps answering questions, e.g. on the duration of breaks or the covered distance and speed of flights, or if they support the detection of activities or other events. Each event has a name. If several similar alternatives of events are necessary, e.g. a pause of 30 seconds as well as a pause of 5 minutes, then the corresponding properties defining each alternative are added as a subscript, for example $pause_{timespan}$. Even though the definition of some events depend on a certain timespan, this attribute is

only mentioned as a subscript if several definitions of the event with different timespans are expected.

If an annotation of the event is not part of the description, but rather a partial result, then the name is formatted as a function and includes this annotation as a parameter, e.g. $enteredRegion(R)$ where the region R is one important result of the event. The attached length to $paused(L)$ and other events does not have to be added explicitly if a start and end attribute are available. However, it is added to the table since the length *is* an important partial result of the respective events.

	Name	Start / End	Complex	Attached Data
1	$paused_{timespan}$	✓		
2	$paused(L)$	✓	✓	L: Length
3	$flew_{timespan}$	✓		
4	$flew(L)$	✓	✓	L: Length
5	$flewInCircles$			
6	$risenUp$			
7	$wentDown$			
8	$ateInsect$		✓	
9	$enteredRegion(R)$			R: Region
10	$leftRegion(R)$			R: Region
11	$met(X, Y)$			X, Y: Bat / ID
12	$changedDirection$			
13	$accelerated$			
14	$decelerated$			
15	$flewInSpeedCorridor_{range,timespan}$	✓		
16	$flewInSpeedCorridor_{range}(L)$	✓	✓	L: Length
17	$slowHuntingFlight$	✓	✓	
18	$highTransitFlight$	✓	✓	

Table 5.1: Overview of 'Real' Events

If the timespan of an event is important to its semantics, it is marked in the column "Start / End". The decision whether an event is complex or not initially seems to be rather subjective. All events without a fixed timespan as well as hunting and transit flights are considered to be complex. The distinction is equal to whether an event can be expressed in regular continuous queries in a sensible way.

$Paused_{timespan}$ is the definition of a pause for at least $timespan$ time instants, e.g. $pause_{5min}$. If the timespan is not predefined, then the event $paused(L)$ is signaled after L time instants and L is attached to the event, in this case implicitly by the start and end time of the event. The same pattern holds for $flew_{timespan}$ and $flew(L)$.

A *pattern* is the description of an event which is comprised of several other events. It states temporal relations, e.g. sequential, parallel and alternative processes, as well as temporal conditions, e.g. two sequential events have to happen within a certain timespan in order to match the pattern.

It was observed that after a bat successfully ate an insect it flies straight up and circles while eating its prey. The whole process also includes flying to the ground. A simple pattern consists of the events $wentDown$, $risenUp$ and $flewInCircles$. This simple pattern for $ateInsect$ happens without a fixed timespan and is therefore considered to be complex.

The events $enteredRegion(R)$ and $leftRegion(R)$ depend on the position of the tracked object. Both are not considered to be complex although they require memory if soft discretization should be applied, i.e. if flying at the border of a region should not fire events of entering and leaving a region.

The event $met(X, Y)$ can be provided by position streams and additionally by meeting streams consisting of encounters of mobile sensor nodes. The stream generated by positions can be compared to the output of mobile sensor nodes for evaluation purposes.

In order to describe $slowHuntingFlight$, $highTransitFlight$ and more, the events 12 to 16 are proposed. $ChangedDirection$ indicates that the direction of movement changed, e.g. turning to a suspicous noise instead of continuing a straight trajectory. The acceleration and decelaration of the movement speed are covered by $accelerated$ and $decelerated$. $FlewInSpeedCorridor_{range,timespan}$ events are emitted, if the movement speed is within a certain range during a fixed timespan. The event $flewInSpeedCorridor_{range}(L)$ means the same without posing a restriction on the timespan.

The events in table 5.1 can be categorized by the marks at the column “Complex”. All events not marked either have a fixed timespan or only happen at one time instant. They are considered to be implemented easily with common continuous queries. All complex events have a variable timespan and even if an implementation with continuous queries may be possible, it is considered to be at least hard to write. In order to simplify modeling of those complex events, a different notation will be proposed.

5.2 Deriving Abstractions

The introduced events and activities to model bat behavior have different requirements and need to be defined on several abstraction layers.

In order to detect simple events, i.e. events that were not marked as complex in table 5.1, regular continuous queries could be used. For example, *enteredRegion*(R) without soft discretization can be implemented as follows: At first, a stream to test whether a position is inside a region is created for every region. Those streams can be merged together into one stream, which produces results similar to table 5.2(a). Finding the regions that were entered only requires to compare region test results of the last two time instants.

```
1 create stream EnteredRegion as
2 select tr2.time, tr1.region, tr1.batID
3 from TestRegion tr1 [range 2s],
4      TestRegion tr2 [range 1s]
5 where tr1.time < tr2.time
6   and tr1.batID = tr2.batID
7   and tr1.region = tr2.region
8   and tr1.type = "out" and tr2.type = "in";
```

Listing 5.1: Describing *enteredRegion*(R) With Hard Discretization in CQL

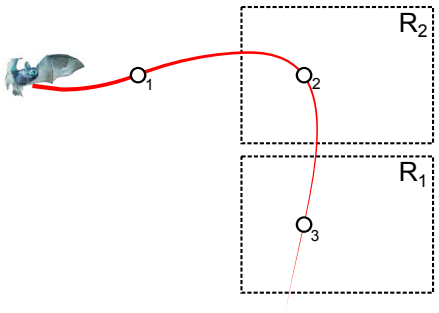
Listing 5.1 is a sample implementation for *enteredRegion*(R) in CQL assuming region tests already exist and are merged as the stream `TestRegion`. For each time instant, `TestRegion` determines whether the current position of a tracked animal is within a region. A sample result of region tests is depicted in table 5.2(a). It is restricted to a bat with the ID 42. The two regions R_1 and R_2 are tested. The behavior of the bat provoking all sample outputs is depicted in figure 5.2(b). The event $leftRegion(R)$ can be created by using the same query and swapping the test results “in” and “out”. Table 5.2(c) shows the result of the query for *enteredRegion*(R) and table 5.2(d) contains the result of $leftRegion(R)$.

Clearly, there are events which can be expressed by common continuous queries. From now on, those will be called *low level events*. They are implemented as continuous queries and the corresponding output streams are interpreted as events.

However, not all events can easily be expressed as a common continuous query. For example, the event *ateInsect* should detect that a bat successfully hunted and ate an insect. A known pattern is that after hearing a suspicious noise the bat lands on the

time	type	region	batID
1	out	R_1	42
1	out	R_2	42
2	out	R_1	42
2	in	R_2	42
3	in	R_1	42
3	out	R_2	42

(a) Sample Merged Region Test



(b) Behavior of Observed Bat

time	region	batID
2	R_2	42
3	R_1	42

(c) Sample Output for $enteredRegion(R)$

time	region	batID
3	R_2	42

(d) Sample Output for $leftRegion(R)$

Figure 5.2: Sample Outputs for $enteredRegion(R)$ and $leftRegion(R)$ Streams and Events

ground, grabs the insect, flies straight up and circles while eating its prey. Table 5.1 contains the events $flewInCircles$, $risenUp$ and $wentDown$ which are now assumed to be already implemented. The actual situation can be described by a simple concatenation of events, i.e. $ateInsect ::= wentDown \triangleright risenUp \triangleright flewInCircles$.

With actual data, this most probably is no strict concatenation due to waiting time between $wentDown$ and $risenUp$ and due to imprecise data or event definitions. Moreover, a timespan is not necessary for the mentioned events and if that is the case there should be waiting time between all three steps. However, adding reasonable waiting times until the next event in the *concatenation* needs to happen is no problem. Listing 5.2 shows a possible query for $ateInsect$ in Esper. The function `timer:interval` allows gaps between events.

```
1 every a=wentDown ->
2         (every (timer:interval(1 min)
3          and b=risenUp(a.batID=batID)) ->
4                 (timer:interval(2 min)
5                  and flewInCircles(a.batID=batID))
6         );
```

Listing 5.2: Describing $ateInsect$ in Esper

From now on, events that can be described by patterns of other events are called *high level events*. Their notation in the prototype is described in the next section.

The last abstraction layer are activities. For each tracked object, there is at, any time, exactly one activity assigned. Activities are similar to a current state of the observed behavior. Hence, UML state charts suggest themselves in order to describe activities. Events may act as transitions between two states. Since the event initiating the transition between two states is considered to be part of the following activity, its start time is used as the start time of the new activity. Figure 5.3 shows an example for a simplified description of activities. Obviously, many activities and transitions are missing. The example nonetheless shows that state charts seem like a convenient notation to describe activities.

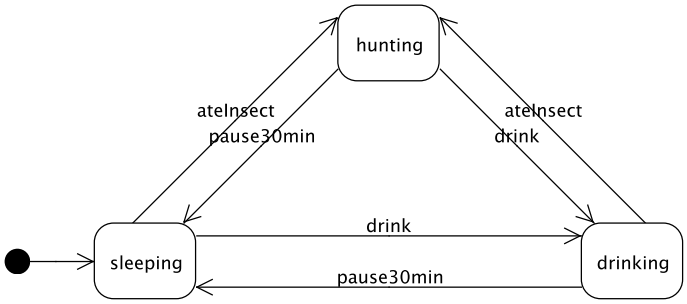


Figure 5.3: Simplified UML State Chart to Describe Activities

State charts still have to be extended by some sort of notation to include uncertainty about the current activity. For example, if a bat decides to sleep, it may not be distinguishable from a short pause at the beginning. The same problem arises at telling apart transit and hunting flights, if the speed is appropriate for both and there was not yet an attempt to catch an insect.

Breaking the rule of only one activity for each time instant could solve this problem. If nondeterminism is included similar to a nondeterministic finite automaton, then all activities that may be correct *are* assigned. If no clear decision can be derived, then alternatives have to be emitted.

A second approach is offering revision transitions. If an event invalidates a previous classification of the current activity, then this activity can be changed to the afterwards hopefully correct one. Revision transitions are only applied to the current state, older history cannot be changed by this notation. Since the decision which notation to choose highly depends on the user, domain experts were asked and opted for revision transitions. In section 5.5 this notation is explained in detail.

5.3 Low- and High Level Events

Since some events to be modeled are more easily defined in a data stream processing approach while others benefit from patterns, both approaches are applied. Hence, this section integrates complex event processing and data stream processing. There are huge similarities to MavEStream (section 3.2.1). Low level events in MavEStream are called *computed events*. Computed events are forwarded to a regular complex event processing engine. This section defines low and high level events and introduces their notation.

5.3.1 Low Level Event

Low level events are regular continuous streams interpreted as events. In order to avoid uncertainties, the streams need to conform to some rules: An event happens at some specific point in time or timespan. If it is just one time instant, then the stream should contain either a *time* attribute or two attributes *start* and *end* with an equal value. If the event definition requires a timespan, then definitely both attributes *start* and *end* are compulsory. Each event has a name and may contain several other annotations, e.g. to include an object ID or spatial information.

If a stream is defined to only generate one event type, there is no need to explicitly add the name to every tuple. Listing 5.3 shows the scheme of a single event definition. `Stream` is either the name of a predefined stream or a select statement. If it is easier to define multiple events in one stream, then this stream has to contain the name of each event in the attribute `type`. The schema depicted in listing 5.4 obviously does not contain an explicit name.

```
1 create Event name as
2 Stream;
```

Listing 5.3: Definition of One Low Level Event

```
1 create Events
2 Stream;
```

Listing 5.4: Definition of Multiple Low Level Events

Both definitions can be interchanged. A named single stream definition can be transformed to one without an explicit name similar to listing 5.5. If all possible

event names for a stream are known à priori, then obviously there is also a converse transformation to several `create event` statements.

```
1 create Events
2 select *, "name" as type
3 from Stream;
```

Listing 5.5: Transformation of Create Event Statements

5.3.2 High Level Event

High level events are patterns over low level events or other high level events. In order to answer biologists questions, several temporal relations between events have to be expressible [All83]. The relations *sequence*, *or* and *parallel* are described in table 5.2. The constant t_{min} in the formal description of the sequence is the smallest time instant possible and is added to the end and chosen timespan in order to guarantee that B happens after A. The expressiveness of the proposed notation is evaluated in section 9.4.1.

Relation	Notation	Informal	Formal
sequence	A ->$_{(timespan)}$ B A -> B	B starts at the maximum of timespan after A A ->(0s) B or $A \triangleright B$	$A.end + t_{timespan} + t_{min} \geq B.start$
or	A + B	A or B happens	
parallel	A || B	A and B overlap	$A.start \leq B.start \leq A.end$ $\vee B.start \leq A.start \leq B.end$

Table 5.2: Event Relations in High Level Events

A notation borrowed from [Espb] is *every*. The pattern *every* (A ->$_{(t)}$ B) matches for every event A followed by a B. After a match was detected, the pattern matcher restarts. Without *every*, the whole pattern matches once. The pattern every A ->$_{(t)}$ every B matches every A which is followed by a B. It may generate more output than *every* (A ->$_{(t)}$ B) if several A events occur prior to a B. For A ->$_{(t)}$ every B the first A is matched to all following events B.

The relations in table 5.2 can be compared to those in [AFR+10]. A sequence every A ->$_{(\infty)}$ every B corresponds to A SEQ B and every A ->$_{(0s)}$ every B is A $MEETS$ B. [AFR+10] does not allow other restrictions of timespans between sequences. However, they can be imitated by "window processes" and other relations. An

every A + every B is the same as $A\ OR\ B$. Moreover, every A || every B is equal to $A\ PAR\ B$.

The consumption of events is also possible in order to allow several associated patterns to only fire until one pattern consumed a requested event, for example for $flew$ in listing 5.12. If an event is consumed in the pattern, it is prefixed with a star, e.g. $*A \text{->}_{(t)} B$ consumes an A event.

The schema of a high level event is nearly complete. Conditions should be added in order to allow an easy correlation to tracked objects. Since events in a pattern may need to be referenced, they can be assigned to a variable. Conditions for an event are put in parentheses. The notation is also borrowed from [Espb]: $a = A \text{->}_{(t)} B(a.batID = batID)$ correlates to events A and B only if their attributes $batID$ are equal. The syntax for finally creating an event is depicted in listing 5.6. Attributes also reference events in a pattern by its variable name.

```
1 create Event
2 name(attribute1, attribute2, ...) ::= pattern;
```

Listing 5.6: Definition of a High Level Event

Since Esper provided a huge amount of the presented notation, it might as well replace the pattern matcher in the prototype. However, output has to be rerouted back into the detection of activities after a replacement.

5.4 Terrain Features

The analysis of position data streams obviously has to take the environment into account. A *special area* might be a pond or a river as well as an area in which a certain type of tree grows. A *region* is a user-defined area of interest. Rectangles, circles and polygons can be used for the specification. Moreover, a region can consist of two regions R_1 and R_2 combined to $R_1 \cup R_2$, $R_1 \cap R_2$ or $R_1 \setminus R_2$. Domain experts have to decide, which regions need to be defined and what their semantics are. Matching positions to regions allows to incorporate contextual information into events, for example a pattern of a drinking bat may use the entering and leaving of a pond region.

Technically, it is not necessary to add any notation or operator. Regular stream definitions already allow selection predicates that resemble region tests. However, manually

writing a region test for a polygon is nontrivial and just invites errors. Hence, extending the notation with regions clarifies queries and increases usability.

Figure 5.4 sketches what a naive approach should return. This can be implemented by creating a stream resembling a region test. A window of two time units compares the results of the region tests and changes yield *enter* and *leave*. This implementation assumes that there are neither holes[1] nor semantic gaps[2] in the position data stream. If a mobile node initially is inside a region, no *enter* is emitted. From now on, this approach is called *hard discretization*.

If there really are no holes and position data is assumed to be exact, this approach is not far from perfect. In reality, there are measuring inaccuracies and a mobile node moving at or near the border of a region yields many arrival and departure events.

Soft discretization adds a tube around the border in which it is unclear whether a point is inside a region. Within the tube, the previous state remains unchanged. Crossing the inner border of the tube means that a point really is within the region until the outer border of the tube was crossed. Figure 5.5 shows the resulting messages of the previous example with soft discretization. Moving at the border now results in fewer messages and measuring inaccuracies are less hurtful. The approach resembles *Enter* ($DS^0 \succ \{DS^B | DS^N\} \succ DS^I$) and *Leave* from [BBBB10].

Listing 5.7 shows the notation of activating *enter* and *leave* low level events in the prototype. The function takes a whole position stream as input and creates a stream adhering the definition of low level events. The input stream is required to provide the attributes x and y. The following functions are defined:

- $ENTERLEAVE(Stream)$: Both *enter* and *leave* events
- $ENTER(Stream)$: Only *enter*
- $LEAVE(Stream)$: Only *leave*

Of course it is possible to add other types of spatio-temporal predicates similar to [BBBB10] in the future.

```
1 create Events
2 ENTERLEAVE(Positions);
```

Listing 5.7: Activating Detection of *enter* and *leave*

1 Missing data in a movement track due to hardware or software malfunction (section 3.4, page 21)
2 An interval in which data is missing on purpose (section 3.4, page 21)

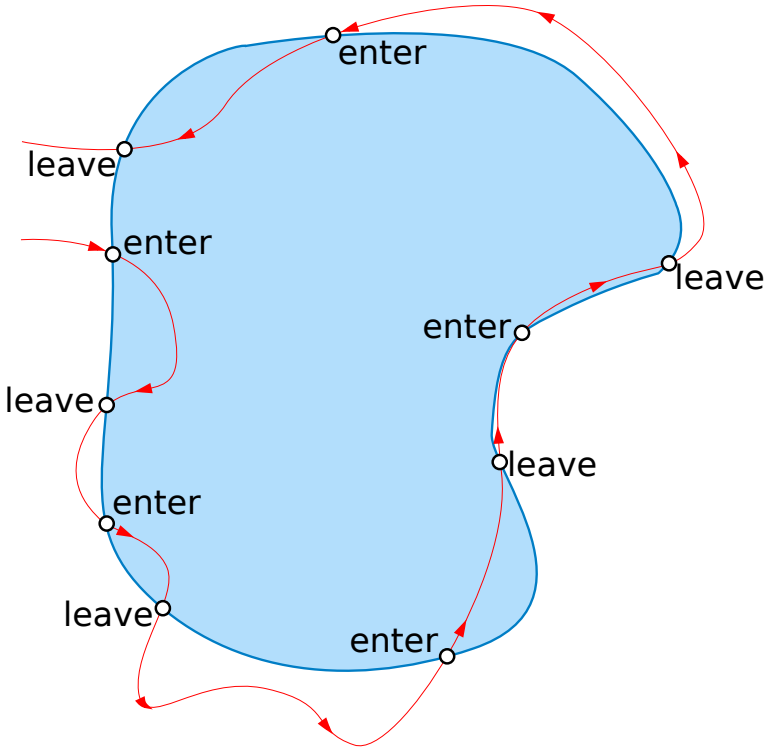


Figure 5.4: Entering and Leaving Regions (Hard Discretization)

Hard discretization is also available in order to just test whether a position is in a region. The following functions are available:

- $getRegions(x, y)$: $float \times float \mapsto \{String\}$
  Returns all matching regions.
- $inRegion(x, y, "R")$: $float \times float \times String \mapsto bool$
  Simple region test depending only on the passed position. The function can be implemented using $getRegions$: $inRegion(x, y, "R") = "R" \in getRegions(x, y)$

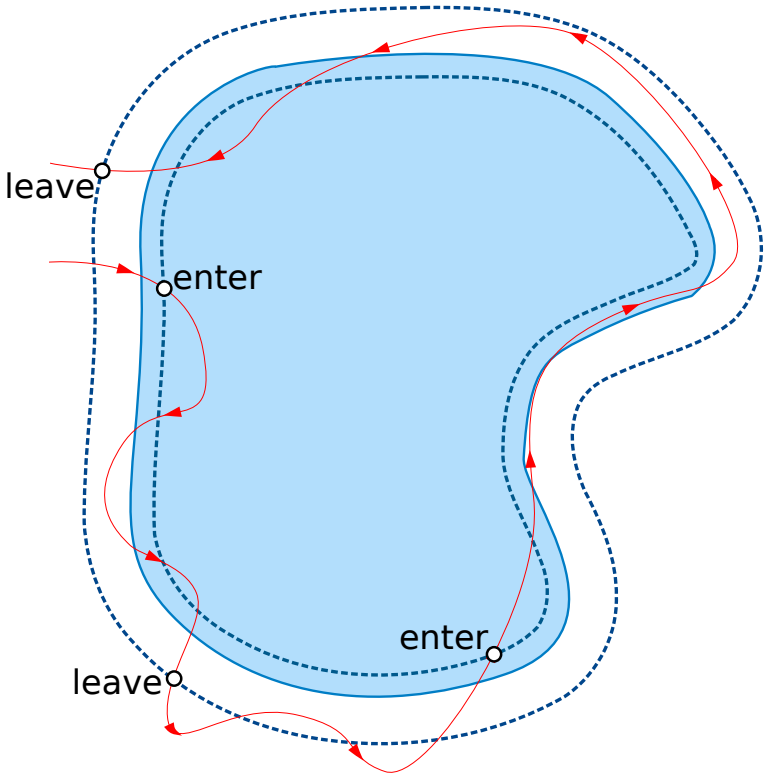


Figure 5.5: Entering and Leaving Regions (Soft Discretization)

5.5 Aggregating Activities

For BATS and related scenarios, there should be certain restrictions on activities. *Each tracked animal is assigned one activity at any time.* If the current activity is unknown, for example because of a lost connection, a backup activity for 'Unknown' should be assigned. Hence there is at least one activity. For this scenario, more than one activity at the same time does not make sense and instead of adding value it would only increase complexity. All modeled bat activities are mutually exclusive, e.g. drinking and sleeping at the same time is not possible. Forbidding parallel activities simplifies predictions and enables modeling similar to state charts.

Furthermore, activities have to be assigned to a tracked object. Events do not necessarily need that, e.g. visibility of the moon does not require to track a dummy 'Gaia' object. Also, a change of an activity means actually changing the activity, e.g. 'foraging' after 'foraging' should be replaced by a longer foraging session.

An essential advantage of modeling activities and transitions between them as UML state diagrams is that it enables to include high level events as simple triggers for

transitions. It abstracts away complex patterns for high level events depending on the previous state and unmaintainable criterions for exclusion for each activity event.

For better applicability, those state diagrams need to be extended. At runtime, activities may need to be revised if they were not properly detected. For example, in the first 30 minutes of resting there is no difference between making a break and sleeping. Just flying around in the woods and foraging also is actually the same until a bat tries to catch an insect or displays known flight patterns of hunting or transit.

Of course, this problem could be hidden away by waiting for the next activity change and applying a filter to results. Unfortunately, this approach would prevent live reports of current activities of observed animals. In section 5.2, a nondeterministic approach as well as UML state diagrams extended with revision transitions have been sketched. Since the decision which notation to choose highly depends on the user, both approaches were presented to domain experts. They opted for *revision transitions*. Revision transitions replace the last chosen activity with a revised one. The start time of the revised activity is the start time of the replaced activity.

Hence, introducing revision transitions also enables the use of dummy states to simplify otherwise complex conditions or patterns of activities. Those dummy states do not need regular transitions, hence they never show up in final activity overviews. The final extension to the state diagrams are *aliases* which simplify the usage of dummy states. A state can be declared an alias of another state in order to just have the same name and the guarantee of true activity changes, e.g. a dummy state for pause does not follow another pause, they are combined. Since at each time instant only one activity is valid, a state can only be an alias for at most one other state. One state may have as many aliases as necessary. Aliases both allow sane overviews of current activities and models similar to hidden Markov models, whereas aliases are observable evidences and dummy states the hidden actual states.

Figure 5.6 shows all elements of the UML state diagram extended with revision transitions and aliases.

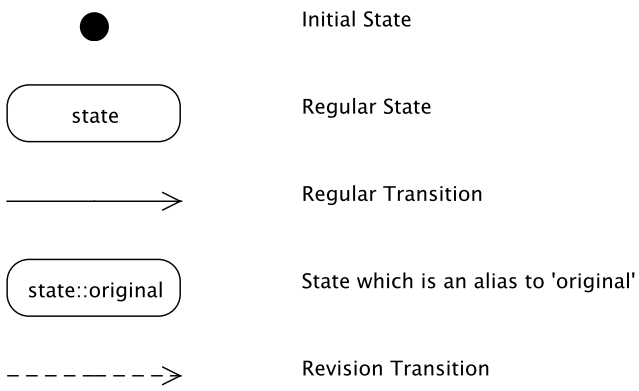


Figure 5.6: Elements of the Extended UML State Diagram for Activities

5.6 Examples

In order to show that the proposed notation really allows to define necessary events and activities, some examples are modeled in this section.

$Paused_{timespan}$ and $Flew_{timespan}$

A simple approach to test whether an animal is currently pausing or in transit is to compare subsequent positions. Considering imprecision of position data, small errors should not result in a wrong interpretation. Furthermore, if a bat makes a pause hanging on a tree, the tree itself might be swinging depending on weather conditions. Hence, small movement should still count as a pause. The following requirements have been elaborated for $pause_{timespan}$:

1. The whole movement track over the period of *timespan* should not exceed a fixed length of $d_{max}^{timespan}$.
2. A fixed amount of subsequent positions should not exceed a (smaller) fixed length of d_{max}^{short}. This allows small errors up to a defined limit and rules out short flights in the observed timespan.

The requirements for $flew_{timespan}$ are similar:

1. The whole movement track over the period of *timespan* has to exceed a fixed length of $d_{min}^{timespan}$.

2. A fixed amount of subsequent positions has to exceed a fixed length of d_{min}^{short}. This rules out a short pause.

It is not required that each time instant contains one of both events, which is the case if there is no signal or if the selected parameters differ. Figure 5.7 sketches the behavior of both events. The classification of the depicted trajectory depends on the chosen limits. Only queries required for $pause_{timespan}$ are outlined since both events are rather similar.

```
1 create stream SubsequentDistances as
2 select p2.time, p2.ID,
3        sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2)) as dist
4 from Positions p1 [range 2s],
5      Positions p2 [range 1s]
6 where p1.time < p2.time
7   and p1.ID = p2.ID;
```

Listing 5.8: $Paused_{timespan}$: Subsequent Distances

Listing 5.8 shows how to retrieve subsequent distances. All examples assume a frequency of 1Hz of incoming data per bat. In listing 5.9 the total length of the trajectory within $timespan = 60s$ is aggregated. A short period of $5s$ was chosen for the second requirement. The stream $TotalDistance5s$ is not explicitly stated, but similar to listing 5.9.

The second requirement has to be true for each $5s$-period within the timespan. Listing 5.10 allows to check this for the whole timespan.

Finally, listing 5.11 combines these streams to a low level event for $paused_{timespan}$.

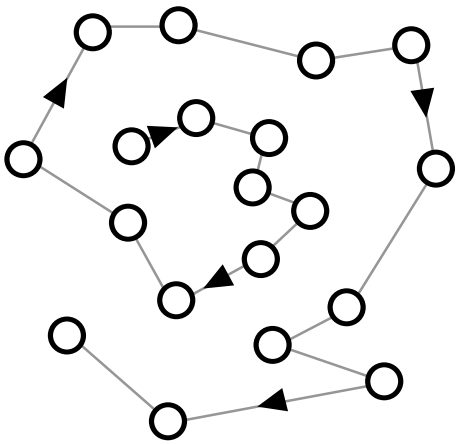


Figure 5.7: Sketch for $paused_{timespan}$ and $flew_{timespan}$

```
create stream TotalDistance60s as
select MAX(time) as time, ID, SUM(dist) as dist
from SubsequentDistances [range 60s]
partition by ID;
```

Listing 5.9: $Paused_{timespan}$: Total Distance Over $60s$

```
create stream Max5sDistance60s as
select MAX(time) as time, ID, MAX(dist) as dist
from TotalDistance5s [range 56s]
partition by ID;
```

Listing 5.10: $Paused_{timespan}$: Maximum $5s$-Distance Within $60s$

```
create event paused60s as
select t.time, m.ID
from Max5sDistance60s m [range 1s],
     TotalDistance60s t [range 1s]
where m.ID = t.ID
  and m.dist < $MAXSHORT
  and t.dist < $MAXLONG;
```

Listing 5.11: Definition of $Paused_{timespan}$

Flew

It is now assumed that $flew_{60s}$ has already been defined and its definition includes an annotation *dist* for the total distance of the trajectory. Similar to $paused_{timespan}$ it has only one time instant. A simple approach to $flew$ is depicted in listing 5.12. The first definition of $flew$ has to explicitly state the start of the event. The attributes *time* and *end* are implicitly assigned.

```
create event
flew(b.ID, (a.dist + b.dist) as dist) ::=
every a=flew ->(75s) *b=flew60s(a.ID = ID and (time - 59s) > a.end);

create event
flew(a.ID, (a.time - 59s) as start, dist) ::=
every *a=flew60s;
```

Listing 5.12: Definition of *Flew*

The timespan between two subsequent flights to be interpreted as one $flew$ event is not allowed to exceed $15s$ in this example. In order to reduce parallel `flew` events,

`flew60s` are consumed: If the pattern matcher has not seen a current `flew` event when `flew60s` arrives, a new `flew` is created. Otherwise, the event is consumed and the pattern matcher does not arrive at the second pattern.

EnteredRegion(*R*) and *LeftRegion*(*R*)

Listing 5.7 in section 5.4 already defined *enter* and *leave* for regions with soft discretization.

$FlewInSpeedCorridor_{range}(L)$

$FlewInSpeedCorridor_{range}(L)$ is similar to $Flew$. In order to enforce the speed corridor, the event `flew15s` is filtered for speed. Listing 5.13 depicts this filtering. A smaller timespan was chosen to allow more precise speed measurements.

```
1 create event
2 flew15sCorridor15_25(a.ID, (a.time - 14s) as start) ::=
3    every a=flew15s(dist > $MIN and dist < $MAX);
```

Listing 5.13: $FlewInSpeedCorridor_{15-25km/h}(L)$: Observed Timespan of $15s$

Unlike $flew$ the distance is not important in this example. Listing 5.14 finally outlines the definition of $flewInSpeedCorridor_{15-25km/h}(L)$.

```
1 create event
2 flewInSpeedCorridor15_25(b.ID) ::=
3 every a=flewInSpeedCorridor15_25 ->(15s)
4          *b=flew15sCorridor15_25(a.ID = ID);
5
6 create event
7 flewInSpeedCorridor15_25(a.ID) ::= every *a=flew15sCorridor15_25;
```

Listing 5.14: Definition of $FlewInSpeedCorridor_{15-25km/h}(L)$

AteInsect

The definition of *ateInsect* depends on the availability of the z-axis and actual bat behavior. Figure 5.8 sketches a successful hunt for an insect after catching it. The whole pattern includes landing on the ground after hearing a suspicious noise. On the ground,

the bat grubs for the suspected beetle. After successfully catching the insect, the bat flies straight up and circles in the air while consuming its prey. Section 5.2 sketched the pattern $ateInsect ::= wentDown \triangleright risenUp \triangleright flewInCircles$.

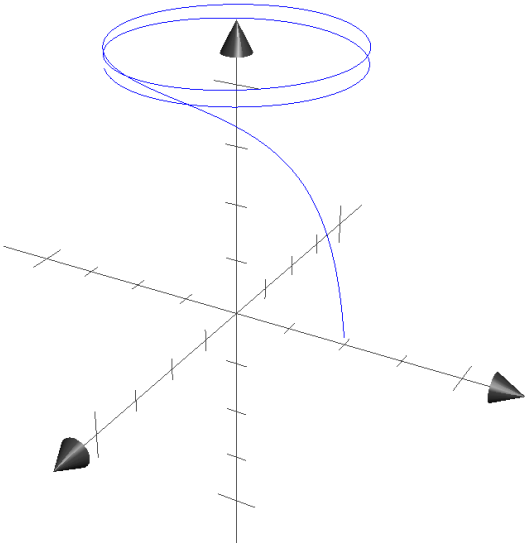


Figure 5.8: Sketch of Successful Hunting Behavior

If no z-axis is available, then the definition obviously cannot use *wentDown* and *risenUp*. However, a short pause interrupting a flight followed by $flewInCircles$ can be detected. The final definition of *ateInsect* should certainly be elaborated after actual sensor data has been retrieved. The following paragraphs assume that the z-axis *is* available.

A simple solution to *wentDown* is looking at a small window of for example 10 seconds and comparing the heights of the start and end to predefined limits. The height of the start should be clearly above the ground, whereas the height of the end is not far from zero. The actual limits need to account for the precision of measuring the z-axis in the field as well as possible springy bat behavior. In order to further reduce the influence of errors in measurement, more than one height at the start and end may be averaged. In listing 5.15 three values were averaged and the height at both ends of the interval is compared to predefined limits `$MIN` (start) and `$MAX` (end). The sliding windows possibly produce several *wentDown* events. Unlike *enter* and *leave*, this does not matter for *wentDown*, since all produced events represent the same information.

The inverse event *risenUp* can be defined in the same way with other height limits and is now assumed to be already implemented. The basic approach to define $flewInCircles$ is less straightforward. At each time unit, one position is measured. The radius of the

```
create stream AveragedHeight as
select ID, AVG(z) as height
from Positions [range 3s]
partition by ID;

create event wentDown as
select (a2.time - 9s) as start, a2.time as end, a2.ID
from AveragedHeight a1 [range 8s],
     AveragedHeight a2 [range 1s]
where a1.ID = a2.ID
  and a1.time + 7 = a2.time
  and a1.height > $MIN
  and a2.height < $MAX;
```

Listing 5.15: Definition of *WentDown*

circle is not very big, in this example it is assumed to be $15m$. Plotted measured points of actual trajectories in the field may yield a cloud of positions similar to figure 5.9 due to measurement precision and "long" timespans between two measurements.

Whether a plotted trajectory of a circular flight really resembles a circle can seriously be doubted. Hence, another approach is proposed: Drawing a circle around a cloud of points and checking whether a certain minimum, e.g. 90%, are within the circle. Summarizing the subsequent distances and comparing them to a minimum trajectory length for a flight also allows to distinguish it from a pause.

The whole procedure can be summarized by the following steps:

1. Finding the center of the circle:
   A query to average values for x and y over a certain window, for example $60s$.
2. Retrieving the total number of considered positions: The query in step 1 can be extended with `COUNT()`.
3. Counting all measured positions within the window which are within the radius around the center.
4. Summarizing the total distance of subsequent points within the window.
   This has already been implemented in listing 5.9.
5. Joining the previous streams at each time unit and checking limits for the total distance and the counted points around the center.

Since all single steps are trivial, they are now assumed to be implemented. All necessary events to describe *ateInsect* are available. The timespan between `wentDown` and `risenUp` are now assumed to be a maximum of $45s$ to grub for a beetle. Even though flying in

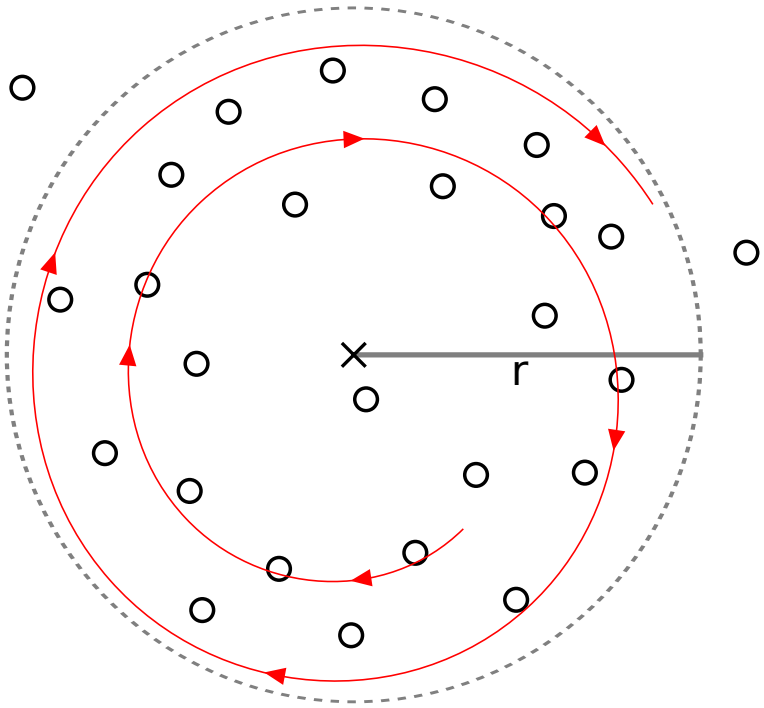


Figure 5.9: Sketch of Expected Measured Data and Detection

circles should start immediately, a safety margin of 10s is applied. Listing 5.16 contains all depicted timespans and the elaborated events.

```
1 create event
2 ateInsect(down.ID) ::=
3 every down=wentDown ->(45s)
4    risenUp(down.ID = ID) ->(10s)
5       flewInCircles(down.ID = ID);
```

Listing 5.16: Definition of *AteInsect*

Activities

The following examples introduce the usage of revision transitions and aliases. Figure 5.10 distinguishes sleeping and a pause. After 30 minutes of no movement, the previous detection of a pause is replaced with the activity 'sleeping'.

The next example in figure 5.11 is still only a simplified model of bat behavior. However, it includes a dummy state. Table 5.3 shows how this model behaves for different events. The subscript after events states the time instant, at which the event occured. The

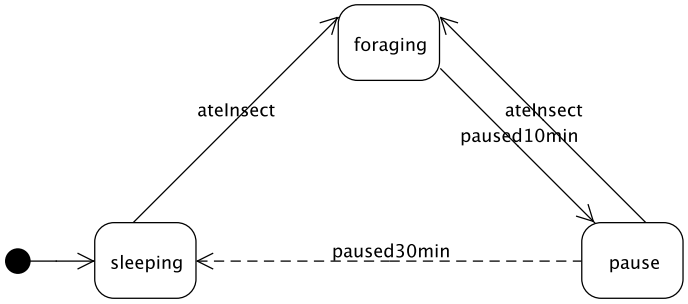


Figure 5.10: Simplified Model Distinguishing Sleeping and a Pause

subscript after resulting activities marks the start time of the activity. The revision allows short breaks while foraging. As long as the dummy state foragingbreak is not replaced, the visible state to the user is pause. Emitted activities contain a *preliminary*-flag in order to specify whether the activity can be revised. Input 2 in table 5.3 is caused by two subsequent foraging activities merged to one.

Figure 5.12 shows the whole model of bat behavior. It uses a dummy state for transit, drinking and hunting which also is used as the initial state. Short breaks during hunting sessions are emitted as a pause. Restarting to fly directly changes the activity back to the hunting session.

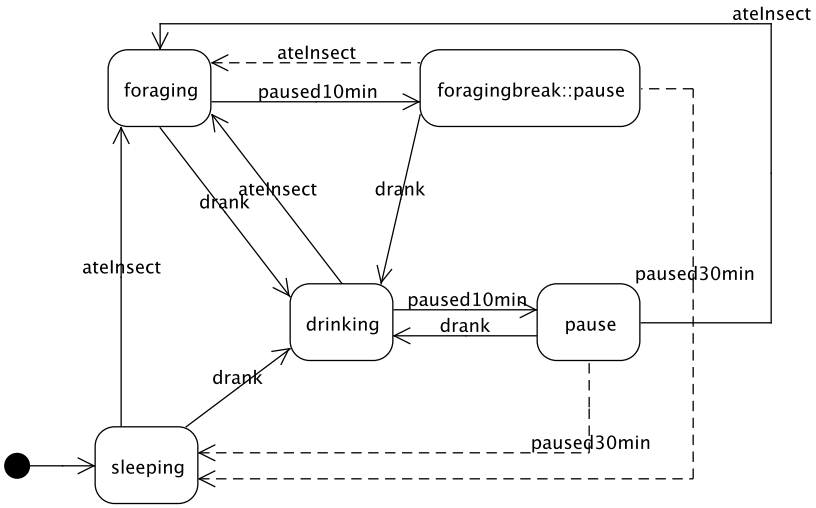


Figure 5.11: Simplified Model Allowing Short Breaks While Foraging

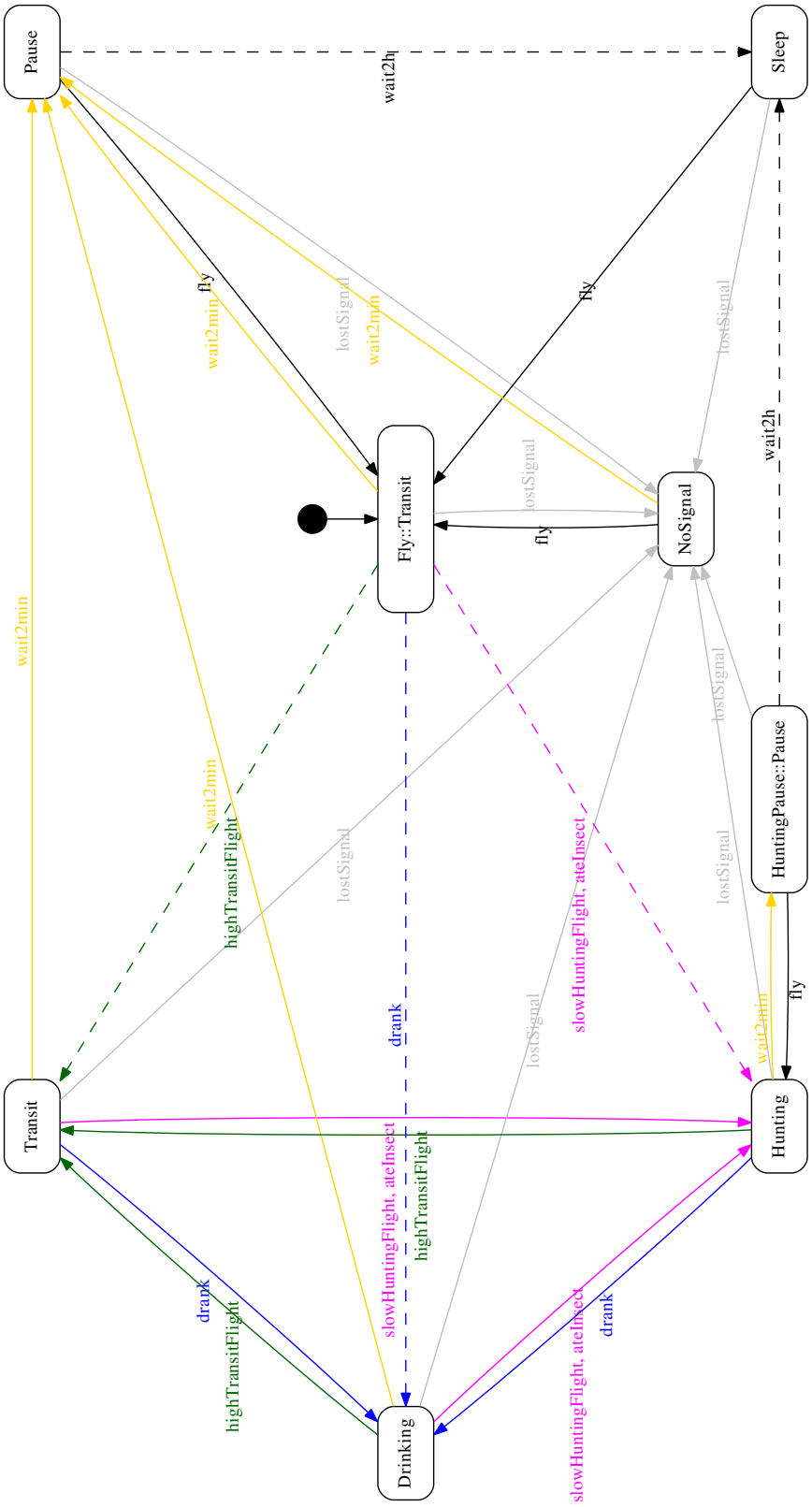


Figure 5.12: Possible Model to Describe Bat Behavior

	Input Events	Resulting Activities
1	-	sleeping_0
2	ateInsect_1	sleeping_0, foraging_1
3	ateInsect_1, paused10min_{650}	sleeping_0, foraging_1, pause_{650}
4	ateInsect_1, paused10min_{650}, drank_{900}	sleeping_0, foraging_1, pause_{650}, drinking_{900}
5	ateInsect_1, paused10min_{650}, ateInsect_{800}, drank_{900}	sleeping_0, foraging_1, drinking_{900}
6	ateInsect_1, paused10min_{650}, $\text{paused30min}_{1850}$	sleeping_0, foraging_1, sleeping_{650}

Table 5.3: Examples of Detected Activities in Figure 5.10

5.7 Summary

This chapter introduces the terms *event* and *activity*. Their detection allows biologists to test hypotheses on animal behavior and to draw new conclusions.

Motivating behavior to be modeled and detected in BATS is introduced. Requirements for a possible notation considering different abstraction levels of behavior is elaborated. The definition of events is separated into *low level events* and *high level events*. Low level events are regular continuous streams interpreted as events. High level events are patterns over low level events or other high level events. Since both event types represent two different processing models, they entail the integration of data stream processing and complex event processing in the prototype.

In order to model activities, UML state diagrams are extended with revision transitions and aliases. The transitions between activities correspond to low level and high level events. Some characteristic motivational events and activities are modeled in the proposed notations.

The definition of activities enforcing exactly one activity for each time instant and transitions between activities allows to introduce state-of-the-art techniques for activity predictions.

6 Predictions

The last chapter introduced activities. Every tracked individual only performs one activity at a time. A sequence of activities represents the behavior during the observed interval. The behavior may depend on observable influences, for example time, brightness and previous activity.

This chapter introduces predictions. Predictions allow to directly exploit an enhanced understanding of the behavior of bats. The presented techniques may be used to answer stated and future questions of biologists as well as supporting experiments. The chosen approach is outlined and available observable influences to be measured are depicted. Moreover, different types of predictions are presented, i.e. not only abstract activities, but also their characteristics.

6.1 Motivation

Being able to predict the behavior of an individual allows to support further observations. A notification in the form of "Bat X most likely is foraging in area Y in 10 minutes" enables biologists to move to area Y themselves or to automatically start already prepared recording equipment which is ready as soon as the bat arrives. It may also trigger not only observing, but also interacting experiments, e.g. emitting certain noises to verify hypotheses on reactions. If an experiment has no access to the public grid, this approach helps to increase the battery life. Even if saving energy is of no concern, it reduces observations and thereby measured data to time intervals of interest.

The techniques of predicting activities can also be used to directly answer several questions of biologists. For example, the hypothesis of whether bats show signs of lunaphobia can be tested by comparing behavior depending on the measured brightness. If behavior is quantifiable and comparable, then a more abstract question on whether bats have a personality can be answered.

Taking spatial characteristics of activities into account, e.g. foraging regions or drinking ponds, allows to check whether individuals frequent previous points of interest and return

to them. Correlations of observable influences to spatial characteristics of activities can also be unraveled.

Finally, the accuracy of predictions is also an indicator of how good the model is. This of course assumes, that changing activities does not happen randomly or only up to a certain degree.

6.2 Dynamic Bayesian Networks

Bayesian networks describe conditional interdependencies between random variables. *Dynamic Bayesian Networks* (DBN) extend Bayesian networks with time and allow more compact representations of hidden Markov models. In order to create a DBN, three types of probability distribution have to be provided: [RN04]

1. Unconditional (initial) probability distribution of state variables X: $P(X_0)$
2. Conditional probability distribution for transitions between states: $P(X_{t+1}|X_t)$
3. Sensor Model for Emissions E: $P(E_t|X_t)$

Activity recognition in the last chapter can be summarized by figure 6.1 from the view of a prediction engine. There is a hidden state of the observed objects as well as the environment. The emissions the prediction engine can see are the output of the techniques presented in chapter 5. Some other emissions for environmental conditions or assumed information on the state of the animals, e.g. whether they had enough to eat, may be revealed in an additional random variable or as payload of the activity random variable. If the state variable is split into more than one random variable, figure 6.1 depicts no longer a hidden Markov model.

The goal of this chapter is to introduce predictions in the form of figure 6.2. The state is splitted into environment information as well as data on the observed animal. The environment as well as an animal has its own state. The environment obviously influences the behavior and thereby the state of a bat. The influence of a single animal

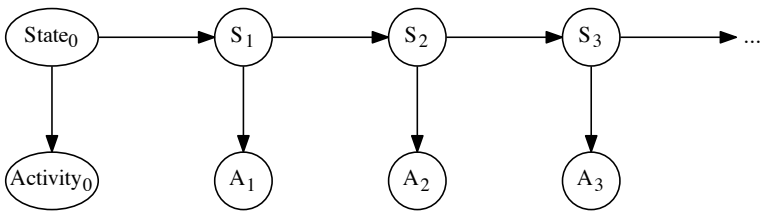


Figure 6.1: Simplified Description of Section 5.5

on the environment may be negligible, but since there is influence, a connection is at least drawn with a dashed line. Activities are emitted based on measured environmental and animal state. Depending on the model, the estimated activity of the last time instant has influence on the next one. Hence, a third sequence chain or rather process was added.

Predictions are based on all available information of environmental and animal state as well as the current activity. Strictly speaking, emission random variables for environmental and animal state are missing. Available emissions will be introduced in the following section and are for now combined with the state variables.

The sketch of the goal of this chapter with Dynamic Bayesian networks already qualifies it to use in predictions. The only missing step are conditional probability distributions between the depicted random variables. After adding them, figure 6.2 *is* the proposed solution.

In order to simplify the notation of dependencies, regular Bayesian networks are now used. Figure 6.3 sketches the approach: The conditional probability tables (CPT) of past observations of activity transitions are used to predict the next activity. It is assumed, that $P(X_{t+1}|X_t) \approx P(X_t|X_{t-1})$, which is usual for DBN.

The networks are modeled by domain experts. If no sensor provides emissions for a random variable, it is either not usable or a static CPT for the variable itself and other directly dependant variables has to be provided. If each and every used variable in a CPT is available as an emission, it can be updated automatically to gain precision over time. These automated updates are intended as the default use case.

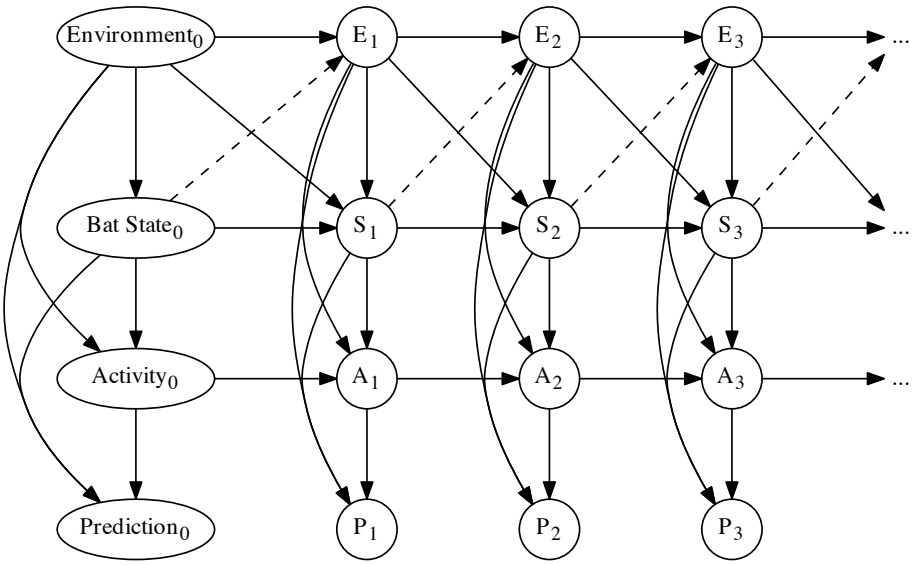
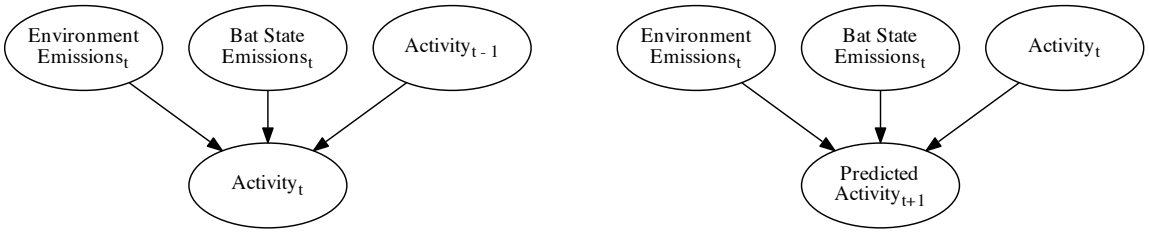


Figure 6.2: Dynamic Bayesian Network Sketching Prediction in BATS



(a) Step 1: Approximate CPT of $Activity_t$ (b) Step 2: Use CPT of $Activity_t$ for Predictions

Figure 6.3: Approximated CPT of $Activity_t$ in Figure 6.3(a) Used as the CPT of $Predicted\ Activity_{t+1}$ in Figure 6.3(b)

6.3 Suitable Random Variables

The behavior of an animal most likely not only depends on its last activity, but on a wide range of environmental conditions and short-term history as well as probably long-term memory. Possible random variables in Bayes networks can be divided into *object state* and *environmental state*. The random variables explained in this section of course can only be a selection of presumably important influences.

6.3.1 Object State

The *current activity* obviously has influence on the next one. Resting may depend on how exhausting a previous hunting session has been and some activity transitions are excluded by definition, e.g. making a break before or after sleeping or equal subsequent activities. The previous, penultimate, antepenultimate or an even older activity probably also still has influence on current behavior. If a longer history is necessary, then the composite Dynamic Bayesian network no longer resembles a first-order Markov model similar to figure 6.2. This may have impact on future optimizations.

The short-term properties *saturation* and *thirst* may also influence the behavior. Saturation may be approximated by counting patterns of successful hunts or summing the lengths of hunting sessions up. Thirst is similar, counting previous drinking events should suffice. Aggregating the counted or measured values to several distinct groups allows to provide small CPTs.

Furthermore, the current location can be an indicator of future behavior. Reusing *region* tests suggests itself since domain experts define regions according to their experience or assumptions. Since regions may overlap, either the whole power set of regions or the power set without impossible combinations represents the random variable.

6.3.2 Environmental Data

An important influence is *time*. For BATS, biologists would partition time according to figure 6.4 in order to produce smaller but meaningful CPTs. A night can be partitioned in several intervals. Too many partitions produce huge CPTs without significant differences for nearby cells in a line whereas too few ones do not add much value. The perfect number of partitions should be determined on real measured data in the field.

Other possible random variables are *weather*, e.g. whether it is raining or exceptionally windy. *Temperature* and *humidity* also deserve their own random variable. In order to test hypotheses on lunaphobia, sensors should be attached to ground nodes in order to measure *brightness*.

6.4 Examples

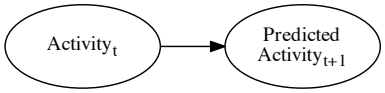
Figure 6.5(a) shows almost the simplest of all possible Bayesian networks to predict the next activity, it only considers the current activity. Table 6.5(d) is a sample CPT which resembles figure 6.5(a). In order to restrict the size of the sample CPT, it assumes there are only four activities (hunting, drinking, pausing, sleeping), even though six have been presented in the last chapter. Since the values of all used variables are available, domain experts do not need to provide a CPT. It is built by the system and should converge to the true probability distribution over time. The only smaller network is a single node for the predicted activity$_{t+1}$ without any conditions.

Figure 6.5(b) uses several random variables. Still, one CPT suffices, since all dependencies are modeled between each present random variable and the prediction variable. It uses environmental data (time) as well as object state (current activity, saturation and thirst).

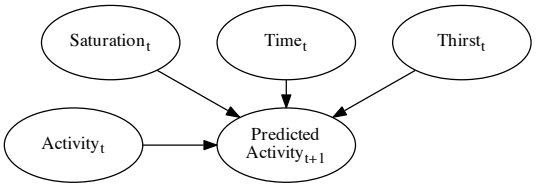
Figure 6.5(c) is a more realistic version of figure 6.5(b). Thirstiness and saturation of a bat changes during hunting sessions and flights to watering places. Time is a restriction on the previous length of hunting and watering flights. Throughout the night, more and more needs are fulfilled and therefore, saturation and thirstiness depends on time.

Day	Night_1	Night_2	Night_3	Day
sunset $+1h$			sunrise $-1h$	

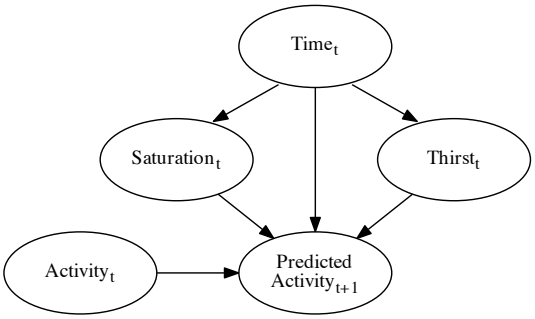
Figure 6.4: Sample Partitioning of Time



(a) Simple Bayesian Network



(b) Several Dependencies



(c) Realistic Dependencies Between Random Variables

	Predicted Activity$_{t+1}$			
Activity$_t$	Hunting	Drinking	Pausing	Sleeping
Hunting	0.0	0.3	0.5	0.2
Drinking	0.6	0.0	0.2	0.2
Pausing	0.7	0.3	0.0	0.0
Sleeping	0.9	0.1	0.0	0.0

(d) Possible CPT of Figure 6.5(a)

Figure 6.5: Examples of Bayesian Networks Predicting the Next Activity

However, the predictions of figure 6.5(b) and figure 6.5(c) are the same. All values of random variables can be approximated. Without missing information, one CPT suffices to predict the next activity and both examples share the same table. Nonetheless, figure 6.5(c) is a better model, since the CPTs of *Saturation* and *Thirst* are also built and updated by the system. The tables are available for further analyses, for example testing *how* saturation changes throughout the night.

6.5 Other Types of Predictions

The presented approach to predict activities can also be applied to their characteristics. Two main attributes of interest are the temporal length of activities as well as their location. Dependencies between random variables are stated as before. Hence, the

structure of the Bayesian network remains the same. The random variable of the attribute to be predicted may be slightly different: If a CPT is not suitable, then a very similar conditional table is used. This chapter states characteristics and employed conditional tables for the attributes length as well as region of activities. Other attributes can be treated similarly.

6.5.1 Length of Activities

There are several approaches to describe a typical length of an activity. The most straightforward way is to use the average. Histograms as well as the median may be helpful in order to identify missing conditions.

Average

Table 6.1 sketches a conditional average table. Instead of a probability its values are the current average. Continuously updating an average is trivial, since the whole history of previous values can be condensed into two intermediate results: the number of values and their sum. The conditional averages can directly be emitted as possible lengths.

		Length of Activity$_t$			
Time	Saturation	Hunting	Drinking	Pausing	Sleeping
Night$_1$	Famished	$90min$	$10min$	$15min$	-
Night$_1$	Peckish	$45min$	$11min$	$10min$	-
Night$_1$	Full	$15min$	$12min$	$15min$	-
Night$_2$	Famished	$1h$	$10min$	$13min$	$12h$
	...		...		

Table 6.1: Conditional Average Table

Histograms

Creating a conditional histogram table is also possible. If only one value as a future length should be emitted, then these tables are only useful if a better function than an average to combine the bars of the histogram is stated, since the proposed conditional average table is more precise. Table 6.2 is an example of a conditional histogram table.

However, a histogram allows to better display the distribution and is therefore intended for results visible to domain experts. Certain spikes in different locations may indicate that not all important conditions have been modeled.

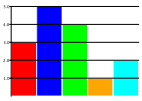
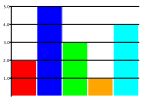
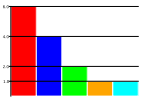
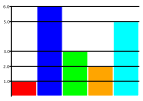
Time	Saturation	Length of Activity$_t$ Hunting	Drinking	Pausing	Sleeping
Night$_1$	Full				
...			...		

Table 6.2: Conditional Histogram Table

Median

The median requires a full history of previous values. It can be approximated by reusing the histogram tables. A conditional median table is also only intended for results visible to domain experts.

6.5.2 Regions

The probability of an event happening in a certain region can be represented by conditional probability tables, since the regions defined in chapter 5 allow discrete random variables. There are several peculiarities to consider:

1. If the activities to be examined contain *all* regions that were visited during the activity, then the random variable should contain the power set of regions.
2. If regions overlap, then the power set of regions in the random variable is necessary. Impossible combinations (no overlap) can be omitted.
3. If regions overlap and all regions visited during the activity are considered, then of course the power set of regions is necessary as well.

Table 6.3 shows a CPT representing region probabilities. It uses the power set of the regions R_1 and R_2, since those two sample events overlap.

Time	Activity$_t$	(Overlapping) Regions of Activity$_t$ $\{\}$	$\{R_1\}$	$\{R_2\}$	$\{R_1, R_2\}$
Night$_1$	Hunting	0.2	0.5	0.2	0.1
...			...		

Table 6.3: Conditional Probability Table for Overlapping Regions

6.6 Adaptive Prediction

The current approach of predicting activities has several disadvantages. Initially, the CPTs and other conditional tables have many empty and unreliable rows. Hence, thorough models initially are rather useless even though there might already be enough information for less precise predictions. Rare cases also do not benefit from well thought-out models due to sparsely filled tables. What happens, if there simply was no past occurence similar to the current situation? Moreover, it was not yet specified whether a conditional table aggregates the behavior of *one*, *all* or a *selection* of observed animals. Until now, a domain expert makes *one* educated guess on the dependencies of random variables. He may have several hypotheses.

Adaptive predictions are proposed in order to eliminate or at least reduce these problems. Domain experts are free to define as many Bayesian networks for activities and attributes as they want.

Even small and imprecise networks not covering all influences are welcome backup plans to use in cases, where precise information is still missing or not reliable enough. Conditional tables can be assigned to the behavior of one animal as well as to a group or all animals. The same Bayesian network can be used several times with different assignments of conditional tables.

The approach can be summarized as follows:

1. All Bayesian networks are updated continuously.
2. All Bayesian networks predict continuously, success is measured.
3. All Bayesian networks are continuously ordered by their success.
4. Emitted predictions:
   Until one network is reliable enough for the current situation, they are checked ordered by their success. If more occurences than a defined limit in the appropriate row of a conditional table are present, then it is assumed to be reliable and the prediction of this network is emitted. Otherwise, the next less successful network is checked.

For discrete random variables, the success measure is the frequency of correct predictions. A success measure for average and median lengths can be an average of eucledian distances between correct and predicted lengths. Measuring success of histograms is possible by comparing its average to the predicted length, by using the probability of the bucket of the actual length or with a user-defined success measure.

In order to measure the reliability of a row, occurences of the row have to be counted. The domain expert decides at which minimum of occurences a row in a conditional table is assumed to be reliable.

Over time, the CPTs gain precision and therefore more thorough models become more successful. Initially, less precise models quickly allow to predict more and more reliably. If different animals display different behavior, then their individual conditional tables also gain precision over time and will be more successful than their aggregated counterpart. Even though initially they may be useless, over time individual conditional tables can therefore even further improve the precision of predictions.

6.7 Summary

This chapter introduced predictions of activities and their attributes. Dynamic Bayesian networks are used in order to consider dependencies between random variables. For the purpose of making the notation of predictions more comfortable, domain experts state Bayesian networks that consist of environmental emissions without any state transition and the transition of animal state. The transition of animal state is represented by random variables for the previous and current activity or respectively the current and predicted activity.

Suitable random variables to be used in Bayesian networks are presented afterwards. Continuous values, e.g. time and temperature, are partitioned for the sake of simplicity. The approach to predict activities can also be applied to their attributes. Hence, peculiarities for two important attributes, the length of an activity as well as its region(s), are presented. Conditional tables for averaged values as well as histograms are suggested.

Finally, adaptive prediction is proposed in order to improve the precision of predictions. Domain experts can now state several hypotheses on the dependency structure of random variables and all corresponding conditional tables in Bayesian networks are updated continuously. The success of predictions is measured and the currently most successful network offering sufficient reliability emits predictions to the user.

The conditional tables in stated Bayesian networks and success measures of adaptive prediction can be used to further analyze behavior. The next chapter presents further analyses leveraging data which was collected to support predictions.

7 Analyses

Providing predictions requires to collect data for several conditional tables in Bayesian networks. These conditional tables can be further analyzed. They allow to make inferences and answer questions on expected activities as well as on whether bats have a personality.

In order to support biologists to interpret findings, visualizations have to be provided. Heatmaps, local convex hulls and activity sequence diagrams are presented in the following sections.

7.1 Heatmaps

There are several types of heatmaps, e.g. regular heat matrices and choropleth maps. Heatmaps allow to relate a multitude of interesting attributes to their position. Of course, the density of positions within a grid cell or region is a useful and common visualization. A visualization of signal strength, velocity and several measured sensor data can be beneficial as well.

Figure 7.1 shows a regular heatmap or rather heat matrix of the velocity of bats. The prototype can also produce thematic choropleth maps depending on region definitions from domain experts.

In order to incorporate changes over *time* into examinations, heatmaps of both types can be combined into an animation. Moreover, to further support domain experts in relating values to terrain features, the prototype optionally overlays satellite images.

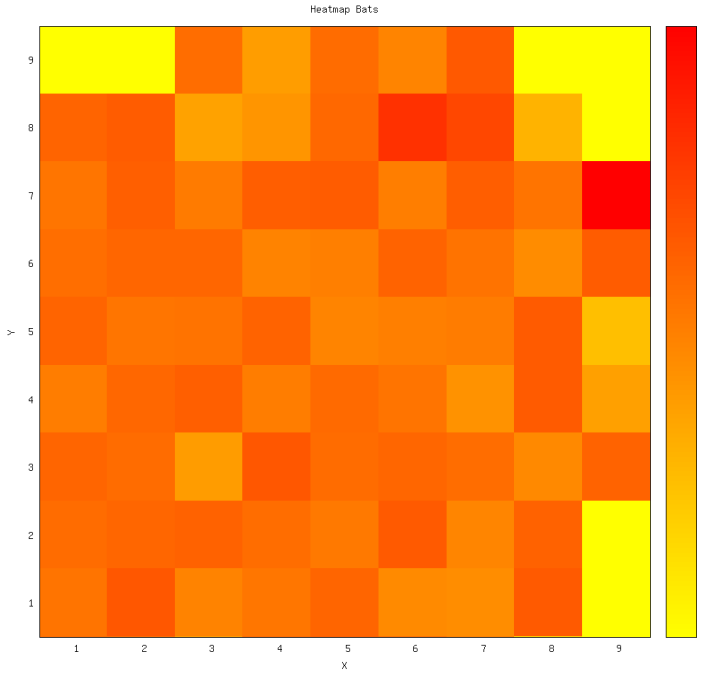


Figure 7.1: Regular Heatmap of Velocity

7.2 LoCoH

Local Convex Hulls (LoCoH) are used to estimate home ranges and utilization distributions. They are able to identify hard boundaries, e.g. rivers or cliffs, and converge to the true distribution as the sample size increases [GFRC+07].

An R script for all three Local Convex Hull (LoCoH) methods introduced in [GFRC+07] is available at [GFR13]. Figure 7.2 is a sample output of the prototype. It depicts a fixed k-LoCoH ($k = 10$) generated from simulated trajectories.

An R script for the newer version T-LoCoH incorporating time is available at [Lyo14]. The prototype provides operators to output fixed k-LoCoH, fixed r-LoCoH, adaptive a-LoCoH as well as the newer T-LoCoH. The parameters k, r and a can easily be adapted in the API. Generating output in several formats, e.g. PDF or SVG, is already prepared and only has to be turned on.

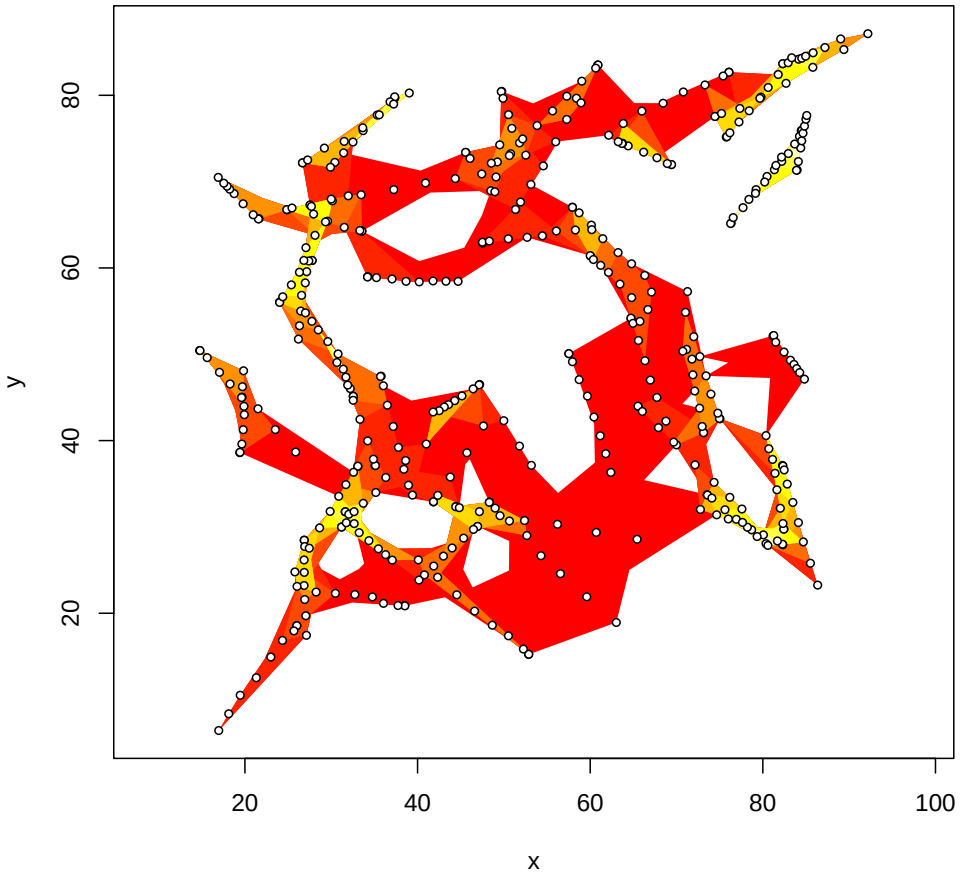


Figure 7.2: Fixed k-LoCoH ($k = 10$) of Simulated Trajectories

7.3 Inference in Bayesian Networks

The last chapter introduced predictions and thereby Bayesian networks into the prototype. The system automatically updates several conditional probability tables as well as conditional tables for regions, averages and histograms. Adaptive prediction also produces a useful byproduct: a success measure of Bayesian networks.

Bayesian networks and conditional tables are not only useful to implement predictions, they can be the fundament for answering several questions. For example:

1. Do bats memorize previous water places and foraging areas and return to them?
2. Do bats show signs of lunaphobia?
3. How do certain conditions, e.g. weather or the current region of a bat, influence their activities?
4. Do bats have personalities?

Chapter 6 is a special case of inference in Bayesian networks. The basic approach can be summarized by the query $P(activity_{t+1} = X \mid activity_t = Y, e_1 = E_1, \ldots e_n = E_n)$, where the current activity Y as well as available emissions $E_1, \ldots, E_n$ are used to calculate the probability distribution of X. Predictions also use and predict attributes of activities.

The general case is $P(Hypothesis \mid Conditions)$, in which every random variable is usable. Conditional tables for regions, averages and histograms are also available. However, conditional average and histogram tables do not contain probabilities and hence the Bayesian theorem does not apply. An output is only possible if there is a conditional table for the random variable V. From now on, using these special conditional tables is described by $Regions(V \mid Conditions)$, $Averages(V \mid Conditions)$ and $Histograms(V \mid Conditions)$.

Answering the first question on whether bats memorize previous water places and foraging areas can be supported: A conditional region table `ActivityRegions` is assumed to be available. It obviously depends on $activity_t$ and probably on other conditions. Creating a table `Regions(ActivityRegions |` $activity_t = drinking$`)` out of the available conditional region table probably yields results similar to table 7.3(a). The regions $R_1, \ldots, R_4$ are distinct water ponds far away from each other. Free variables allow to dig in, an example is X in `Regions(ActivityRegions |` $activity_t = drinking \wedge time = X$`)` which may produce an output similar to table 7.3(b). Both results show clear preferences for certain regions. Foraging areas can be examined in the same way. The resulting tables need to be interpreted by domain experts in order to decide whether they support a hypothesis. In this case, biologists need to assess whether a clear preference of R_1 over a long observation interval can be attributed to memory.

R_1	R_2	R_3	R_4
0.6	0.1	0.05	0.25

(a) `Regions(ActivityRegions |` $activity_t = drinking$`)`

time	R_1	R_2	R_3	R_4
Night_1	0.9	0.05	0.0	0.05
Night_2	0.3	0.1	0.1	0.5
...			...	

(b) `Regions(ActivityRegions |` $activity_t = drinking \wedge time = X$`)`

Figure 7.3: Sample Results of Processing Conditional Region Tables

There is also the question of whether bats show signs of lunaphobia, i.e. whether the visibility of the moon has influence on their behavior. In order to answer this question, the brightness or visibility of the moon has to be available as an emission. The emission has to be included in a Bayesian network for predictions and the random variables for activities and their attributes have to be modeled to depend on the emission. The following conditional tables should be produced:

1. $P(activity_{t+1} = X \mid activity_t = Y \wedge moon = M)$
   Compare transitions of activities or rather expected activities with and without a visible moon.
2. $Regions(activity_t \mid moon = M)$
   Compare the location of activities.
3. $Histograms(activity_t \mid moon = M)$ or $Averages(activity_t \mid moon = M)$
   Compare the duration of activities.

In order to show that the moon *has* influence on the behavior of bats, at least one of the corresponding conditional tables has to display significant differences regarding the visibility of the moon.

All conditional tables can be exported in order to analyze them in suitable external applications.

7.4 Expected Activities

Expected activities are a specialization of inference in Bayesian networks similar to predictions. They are always of the form:

$$P(activity_t \mid v_1 = \varphi_1 \wedge \ldots \wedge v_n = \varphi_n \\ \wedge v_{n+1} \in \{\gamma_{n+1}^1, \ldots, \gamma_{n+1}^{l_1}\} \wedge \ldots \wedge v_{n+m} \in \{\gamma_{n+m}^1, \ldots, \gamma_{n+m}^{l_m}\})$$

All $v_i, i \leq n+m$ are random variables and $\varphi_j, j \leq n$ are either possible values of v_j or free variables. For each $v_{n+k}, k \leq m$ a restriction to l_k possible values $\gamma_{n+k}^s, s \leq l_k$ is applied. The variable for $activity_t$ is always free. The result is a CPT and its conditions are all used free variables.

This definition is similar to predictions, although it does not depend on live emissions. It allows to predict behavior for certain conditions. Missing information leads to more

aggregated results, e.g. $P(activity_t)$ only outputs a probability distribution table with one row.

7.5 Personality of Animals

An approach to answer the question whether bats have personalities is to compare their expected activities for certain conditions as well as attributes of activities to the average behavior.

Domain experts have to decide which conditions for activities and attributes they consider to resemble the personality. Restricting the definition to key characteristics is advised, even though all collected data in conditional tables assigned to individual animals can already be interpreted as the personality. It allows to examine this question in a wide range of modeled Bayesian networks and to easily define groups of behavior, e.g. conservative or wild bats.

Hence, the definition of the personality of animals should consist of several conditional tables stated similar to expected activities:

- $P(activity_t \mid Conditions)$
- $Averages(attribute \mid Conditions)$
- $Histograms(attribute \mid Conditions)$
- $Regions(attribute \mid Conditions)$

Of course, not all types of conditional tables have to be used and several tables of the same type are allowed. Functions to determine the difference between conditional tables have to be stated, since this highly depends on the domain.

If the applied definition is coherent and differences between its conditional tables to the average are significant, then the question of whether bats have a personality can be affirmed.

7.6 Activity Sequence Diagrams

In order to visualize the results of activity detection and prediction, *activity sequence diagrams* are proposed. The axes of activity sequence diagrams are a time line as well as distinct values (*lanes*) for the IDs of tracked animals. Each activity is displayed by a rectangle in the appropriate lane and restricted to the detected interval in the time line.

Events may also be blended into the lanes of the diagram. The diagram is structured similar to activity patterns in [RLH09].

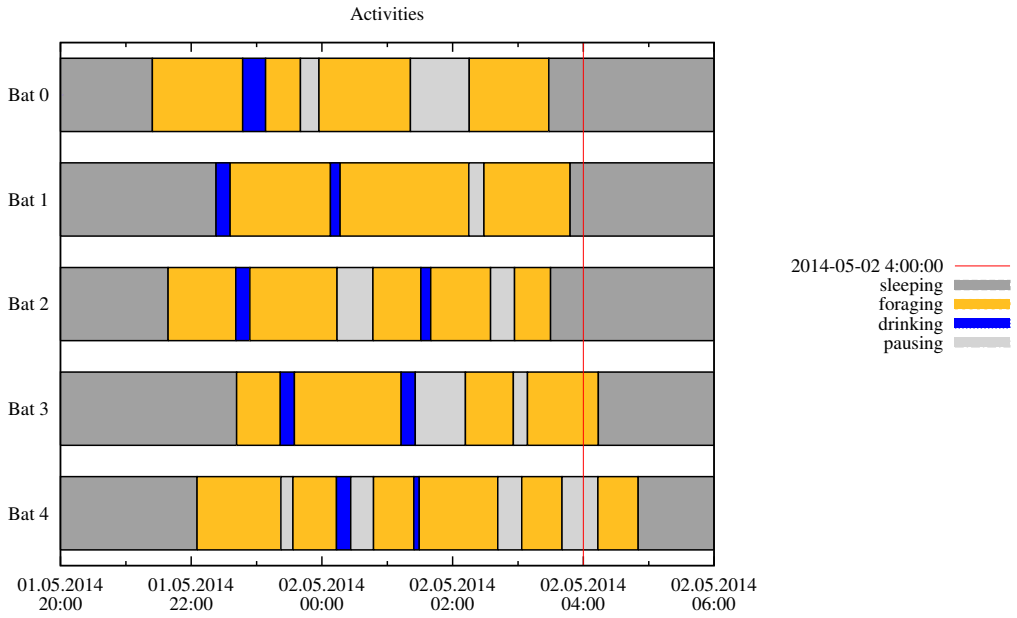


Figure 7.4: Sample Activity Sequence Diagram Presenting a Simulated Night

By inserting a separator, past detections can be distinguished from predictions. Figure 7.4 shows an example of an activity sequence diagram containing all elements of the definition. The separator between past activities and predicted behavior is the vertical red line at the time instant 2014-05-02 4:00. The diagram is an actual output of the prototype. However, the displayed data stems from simulated activities, not detected activities in the field.

7.7 Summary

Heatmaps and local convex hulls are state-of-the-art analyses. Heat matrices and choropleth maps allow to display various types of interesting attributes related to their location. Local convex hulls are used to determine home ranges and utilization distributions.

This chapter also introduced inference in Bayesian networks. A notation for using several types of conditional tables, e.g. regular CPTs as well as conditional average,

histogram and region tables, is proposed. Special cases of Bayesian inference are predictions in chapter 6, expected activities as well as the personality of animals.

Activity sequence diagrams allow to visualize both past sequences of activities and predictions. Their structure is similar to activity patterns in [RLH09].

8 Implementation

The presented analyses in the last chapter already provided some results which were generated by the prototype. They examine characteristics at several abstraction layers, from position data to inference on behavior. This chapter provides an overview of how data passes through the prototype or rather how several techniques are connected in order to enable analyses for different abstractions.

Different stages are explained in more detail, for instance available operators in data stream processing and event detection. Moreover, this chapter describes the integration of predictions of activities into the prototype and other data stream systems.

8.1 Overview

Many techniques and analyses have been presented. Figure 8.1 shows the path data takes through the prototype.

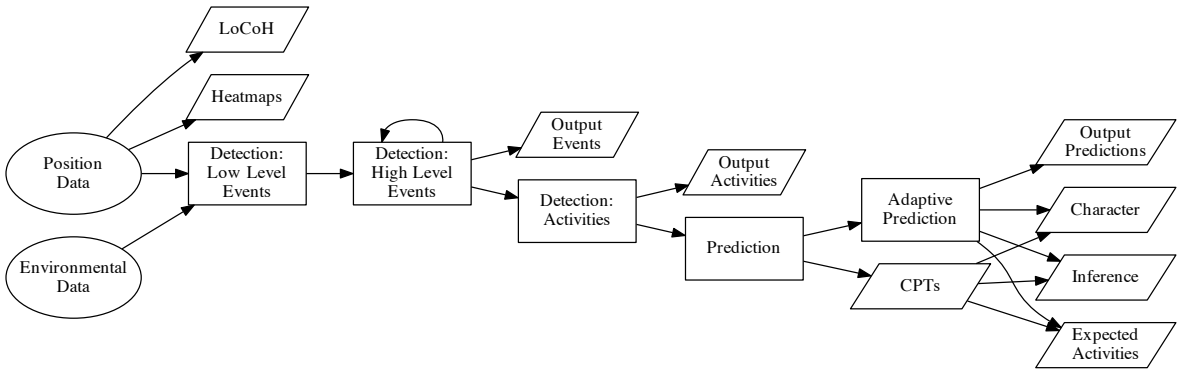


Figure 8.1: Path of Data Through the Prototype

Position data is pushed to external gnuplot and R scripts for heatmaps and local convex hulls, which are both state-of-the-art analyses. Position data as well as additional environmental data are further processed to low level events. Low level events are pushed to the detection of high level events. They are emitted to the user and interpreted as transitions for activity detection. Activities are emitted, too. The history of activities

and their dependencies are aggregated in CPTs. The success of CPTs is measured and the prediction of the previously most successful conditional table is displayed to the user. The CPTs and success rates identified by adaptive prediction provide analyses for the character of animals, expected activities and more generally inference on Bayesian networks.

8.2 Simulator

Since the sensor network montioring basts is not yet deployed, event detection has to be evaluated on synthetic data. The simulation framework STREAMIC [Ste13] was extended with some more realistic behavioral patterns to be detected by the prototype. The main goal is to simulate hunting behavior during the night.

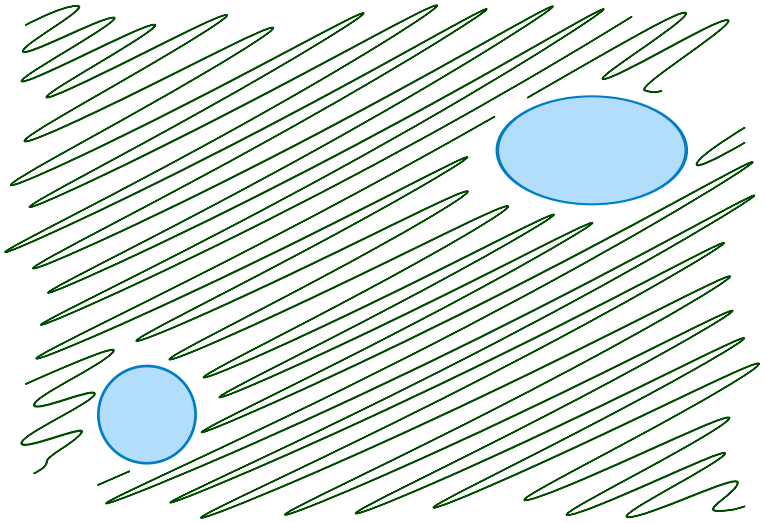


Figure 8.2: Simulated Forest (Sketch)

Figure 8.2 shows the simulated forest. The forest consists mostly of trees, which are depicted as hatched areas. Two water ponds allow simulated bats to quench their thirst.

The original approach of how bats find their food was kept up. Hundreds of simulated insects are randomly spread across the forest. If a bat is hungry, the simulator randomly selects an insect and the bat is heading for it. This was extended by randomly selecting several insects and choosing the most nearby one. In order to include frequent changes of direction, from time to time the target insect is replaced. Moreover, the bat does not always succeed in catching the insect. If it succeeds, the movement track depicted in figure 5.9 (page 59) is applied in which the bat circles in the air while consuming its prey.

The simulator now includes some indicator to control the need of resting and an indicator for saturation. The behavior was extended to rest if necessary, to drink from time to time, to hunt if the bat is hungry and otherwise to randomly follow another bat. Furthermore, the simulator gained values for the z-axis for movement tracks of bats.

8.3 Data Stream Processing

The fundament to data stream processing in the prototype are the semantics of PIPES (section 4.2). All following operators are implemented for chronon streams. Hence, a possible future optimization especially considering large windows is to switch to general physical streams with variable time intervals.

8.3.1 Standard Operators: Projection and Selection

Selection and Projection are implemented in the same way as in [Krä07a] and relational databases. For each tuple, *Selection* checks whether a given predicate is fulfilled. If it is, the tuple is emitted. In PIPES, Selection is called *Filter*. *Projection* applies a mapping function to a tuple in order to emit an altered one. In PIPES, it is therefore called *Map*.

8.3.2 Combining Streams: Union, Cartesian Product and Join

Union merges two streams of compatible types [Krä07a]. Its semantics are also exactly the same as in the relational model.

The *Cartesian Product* "of two logical streams combines elements of both input streams whose tuples are valid at the same time instant" [Krä07a]. The definition of the Cartesian Product in PIPES and in the relational model differ.

Join is an optimization of the Cartesian Product and a following Selection: It combines tuples of both input streams at the same time instant which fulfill a mutual predicate.

8.3.3 Window

The *Window* operator allows to combine tuples of several subsequent time instants. The implementation corresponds to time-based sliding windows ω^{time} in [Krä07a]. It "shifts a time interval of size w time units over its input stream to define the output stream" [Krä07a].

The interval may also be unbounded. For unbounded intervals, results are emitted at each time instant. The general case assumes that infinite memory is available. However, some queries can be optimized. For example the sum and average over values can use immediate results and therefore do not need to be recomputed completely if values are added.

8.3.4 Scalar Aggregation

Scalar Aggregation computes an aggregate from a set of elements in the same time instant, for example the sum of a certain attribute of all tuples. Grouping is also possible and was written as `partition by` in previous examples. With grouping, Scalar Aggregation emits values for each group similar to the behavior of `GROUP BY` in relational databases.

If the Scalar Aggregation operator is applied after an unbounded window, certain optimizations have to be considered due to the fact that actual systems do not have infinite memory.

In order to speed up the generation of heatmaps, an operator `GridAggregate` was implemented. It groups tuples according to their x and y attributes into a predefined grid and applies aggregate functions. Moreover, `GridAggregate` emits aggregates for empty groups.

8.3.5 Terrain Features

The operator `RegionAggregate` provides the same functionality as GridAggregate for user-defined regions, e.g. polygons. Aggregates for empty regions are emitted in this operator, too.

A region test `inRegion`$(x, y, region)$ is available in the Projection operator. Since an animal ID is not part of the signature, it applies hard discretization.

Soft discretization is available in an additional operator. This operator stores a state for each animal and region and decides whether an animal is within a region based on the previous state, the current position and the inner and outer limits of the region.

8.3.6 Real-Time Considerations

The urgency of results depends on the biologists question being asked. Long-term analyses and short-term questions on *current* positions obviously have different temporal requirements. Hence, *Real-Time* in this context does not stand for hard real-time

conditions to be satisfied. Its meaning resembles the term *urgent* and requires to not wait for too long for missing data before answering a question.

The urgency of heatmaps, local convex hulls and inference in Bayesian networks is not very high. The necessary data is collected for several hours or even days. On the other hand, current predictions of activities or even displaying the current position of a certain bat should not wait for unrelated data of another animal in the same stream. Obviously, on simulated data this is not an issue.

In the actual deployed system, an additional operator should therefore be implemented which enforces that the system only waits for a certain amount of time until a tuple indicating missing data is issued and operators are allowed to process the next time instant. The operator should discard tuples which arrive too late.

8.4 Event Detection

The implementation of event detection has been restricted to patterns which are necessary to support all previously introduced events. All event operators work on two events, e.g. $A \mid\mid B$. If event consumption is neglected, binary operators suffice. For sequences, it is assumed that at least the first event is prefixed with `every`.

The restriction primarily shows up at event consumption: It is rarely needed and all examples use no more than one event consumption per pattern. In order to not unnecessarily increase complexity, event consumption has been restricted to binary operators without covering the general case. If this feature is required, either existing CEP systems should be integrated or the implementation has to be extended. The following binary pattern operators are available:

1. every A ->$_{(\delta)}$ B
2. every A ->$_{(\delta)}$ *B
3. every A ->$_{(\delta)}$ every B
4. every A ->$_{(\delta)}$ every *B
5. every *A ->$_{(\delta)}$ B
6. every *A ->$_{(\delta)}$ *B
7. every *A ->$_{(\delta)}$ every B
8. every *A ->$_{(\delta)}$ every *B
9. $A \mid\mid B$
10. *$A \mid\mid B$
11. $A \mid\mid$ *B
12. *$A \mid\mid$ *B
13. $A + B$
14. *$A + B$
15. $A +$ *B
16. *A

With event consumption, the order of patterns is crucial. For each new event, all patterns are called sequentially unless one pattern consumed the event. If a second event to match is still missing, then the pattern adds the new event into a waiting list. Events in waiting lists can be consumed later, too. Hence, if an event in a waiting list was consumed, each pattern is notified to remove that event from its waiting lists.

The `or` operators as well as a single consumption can be useful at the end of a rules sequence to prevent later matches.

8.5 Bayesian Networks in Data Stream Systems

Chapter 6 introduced Bayesian networks and conditional probability tables (CPTs) into the prototype. This section elaborates how they can be expressed in existing data stream systems and how they were implemented in the prototype.

8.5.1 Conditional Probability Tables as Streams

Without loss of generality, table 8.3(a) sketches the goal of implementing the aggregation of CPTs as streams. Table 8.3(b) depicts all necessary streams. One input stream `Input` is available. If actually several input streams are present, they can be joined into one. The input stream has at least two attributes: `value` for the observed output and `situation` for the random variable.

Input	Output	
	A	B
S	$P(A\|S)$	$P(B\|S)$
T	$P(A\|T)$	$P(B\|T)$

(a) Elements of a CPT

$\longrightarrow$

Input	Output	
	A	B
S	**PAS**	**PBS**
T	**PAT**	**PBT**

(b) Streams a CPT

Figure 8.3: Sketch of CPTs as Streams

Each possible output is related to all possible values of the random variable. Hence, in this example there are four output streams: `PAS`, `PAT`, `PBS` and `PBT`. Generally, each cell in a CPT has its own stream.

Listing 8.1 defines the stream `PAT` to approximate $P(A|T)$. It counts the situation S and relates it to the amount of outcomes A for this situation. Obviously, missing previous situations of S yield an empty row and errors. This case has to be treated separately. Some possible strategies are to output a uniform distribution, zero probability or a

value representing *unknown*. Since these strategies can be implemented by simple `IF` expressions and need to be chosen for each scenario, listing 8.1 does not include a strategy and assumes previous situations S. The other streams are formulated accordingly.

```
select COUNT(cell.value) / COUNT(row.situation) AS p
from input as cell [range unlimited],
     input as row [range unlimited]
where cell.situation='S'
  and cell.value='A'
  and row.situation='S';
```

Listing 8.1: One Cell of a CPT

8.5.2 Predictions as Streams

Usually, one CPT should suffice. If the whole network is necessary, for example to cover the case of missing data, then a new CPT can be built which has all available and used random variables as its conditions. Bayes' theorem allows to formulate outputs depending on given input. The continuous queries to build cells for the new CPT join all current cells of necessary CPTs. Hence, without loss of generality, one CPT suffices.

Listing 8.2 assumes a CPT similar to table 8.3(b) is used to predict the output A or B. The most likely output is emitted with its probability or rather belief state. The prediction depends on the current situation, which is provided in an additional stream `input`. Larger CPTs lead to if cascades for predictions, but nonetheless can be implemented similarly.

```
select
   (if input.situation='S'
      then (if PAS.p > PBS.p then 'A' else 'B')
      else (if PAT.p > PBT.p then 'A' else 'B')) as predicted,
   (if input.situation='S'
      then MAX(PAS.p, PBS.p)
      else MAX(PAT.p, PBT.p)) as belief
from input [rows 1],
     PAS [rows 1], PBS [rows 1],
     PAT [rows 1], PBT [rows 1];
```

Listing 8.2: Prediction as a Stream

8.5.3 Actual Implementation

In order to implement predictions with continuous queries, a huge amount of stream definitions are necessary. The actual operations to be implemented are rather simple: update cells or rows in a table and retrieve values of cells in selected rows. Hence, the different data structures are implemented separately. The following tables were implemented:

1. Regular Conditional Probability Tables
2. Conditional Average Tables
3. Conditional Region Tables
4. Conditional Histogram Tables
5. Conditional Median Tables

The signature of all tables is rather similar:

- $addEvidence(Input_1, \ldots, Input_n, Output)$: Update the conditional table
- $predict(Input_1, \ldots, Input_n)$: Predict using previous evidence

Adaptive Prediction requires to measure success and to use the most successful conditional table. Its data structure shares the same signature and passes evidences to its internally managed regular conditional tables.

8.6 External Scripts

Each abstraction of position data, namely streams, events, activities and predictions, allows to execute external applications. For example, heatmaps and local convex hulls are generated by piping data of current time instants into gnuplot and R scripts. Moreover, activities and predictions are visualized by gnuplot in an activity sequence diagram.

The same mechanisms that support visualizations can be used to fire alarms and trigger experiments without the need to continuously examine output streams of the prototype in an external application.

8.7 GUI

A graphical user interface visualizing data from the prototype was implemented with HTML5 and WebGL. WebGL is a cross-platform API used to create 3D graphics in a browser [Khr14b]. Many web browsers already support it, for example Firefox 4.0, Safari 5, Opera 12 beta and Chrome [Khr14a].

This approach was primarily chosen because of its platform independence and since HTML allows to easily format and present various kinds of data. Previous web standards had been supported for a long time, which may render frequent adjustments of the prototype for newer APIs unnecessary.

The visualization uses available javascript libraries `jQuery`[1] for its selector scripts and the WebGL library `three.js`[2]. Bats can be selected by their node on a map and available analyses are listed and linked. Events, activities and predictions are displayed. Figure 8.4 shows a screenshot of the visualization in a browser.

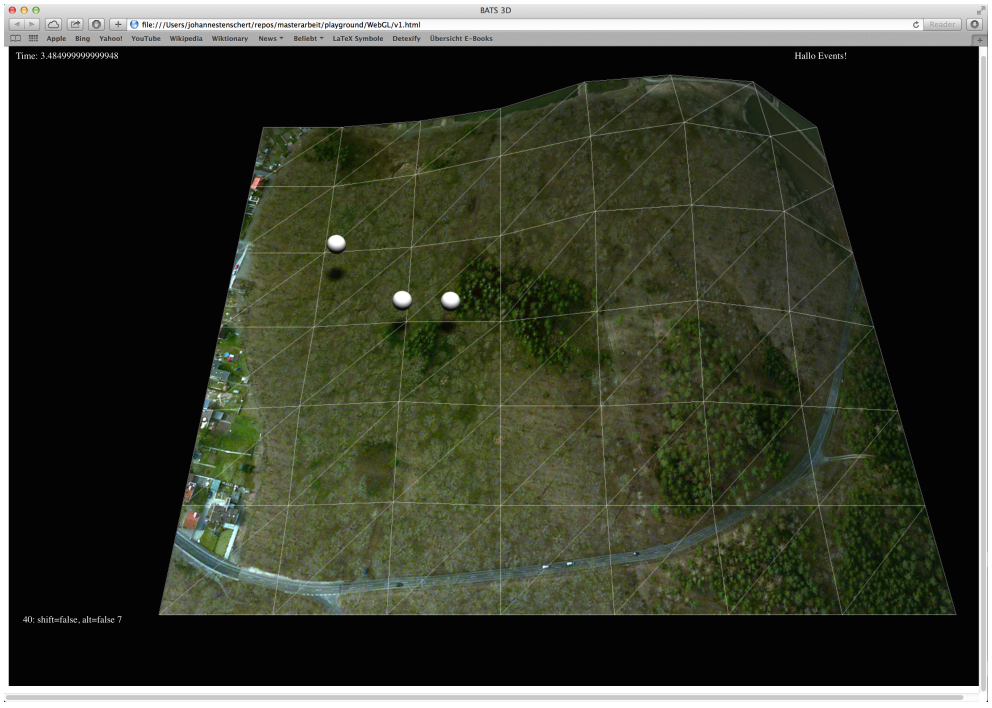


Figure 8.4: GUI Implemented in WebGL

1 `http://jquery.com`
2 `http://threejs.org`

8.8 Summary

This chapter starts with an overview of how data passes through the prototype and thereby how several techniques are connected. Changes to the simulation framework STREAMIC are outlined afterwards. More realistic behavioral patterns were added to be detected in the prototype and to support an evaluation of the prototype and user-defined continuous queries, events, activities as well as further analyses.

The fundament to data stream processing in the prototype are the semantics of PIPES. Implemented operators are explained in detail. Some new data stream operators were added to simplify execution plans in regard to spatial information. Afterwards, available operators for event detection were listed.

Integrating Bayesian networks into data stream systems is explained in detail. Conditional Probability Tables and predictions were expressed as streams. The actual implementation in the prototype is an optimization of the presented approach which updates tables instead of a stream for each single cell in a CPT.

All abstractions allow to execute external applications, for example to trigger experiments or to generate visualizations in external gnuplot and R scripts. Finally, the implemented GUI which uses HTML 5 and WebGL was introduced.

9 Evaluation

This chapter examines the prototype, techniques and notations in regard to several aspects. It is assessed whether objectives are achieved. The impact of adaptive prediction on precision of activity predictions is demonstrated. Detection of events is examined in regard to expressibility of temporal relationships between events as well as actual implemented events. Finally, the influence of data quality on analyses is discussed.

9.1 Objectives

Biologists have several assumptions on the behavior of bats and hypotheses on probable behavior. All expected behavioral patterns can be evaluated: Assumed foraging patterns can be detected and biologists can compare results to their observations or video footage. The questions of memorized water places and foraging areas as well as avoidance of open places can be answered by inference in Bayesian networks and conditional region tables. Whether bats show signs of lunaphobia can also be decided by inference in Bayesian networks. A possible definition of the personality of an animal is sketched in section 7.5

Territorial behavior may be examined by comparing LoCoHs of different bats and other species. The ability to answer questions of the starting age of social activities and whether hunting is taught to a bat by its mother depends on whether very young bats can carry the sensor nodes. If this is possible, events for social activities may be defined and their frequency of detection can be related to the age of a bat. Whether hunting is taught to a bat by its mother can be assessed by comparing activity sequences as well as the spatial distance of a child to its mother.

With execution of external applications, the prototype provides a technique to support experiments. External gnuplot and R scripts of several state-of-the-art analyses were integrated into the prototype. The same mechanisms to call those scripts can be used to easily add new analyses.

Notations for specifying events and activities have been elaborated in chapter 5. Continuous queries can be interpreted as low level events. High level events are patterns over low and high level events. Modeling of activities is similar to modeling state charts.

Events are used for transitions between activities. Hence, modeling techniques for events and activities are provided by the prototype.

Several visualizations are provided:

1. Heat matrices (regular heatmaps)
2. Choropleth maps or rather thematic maps
3. Local Convex Hulls (LoCoHs)
4. Activity Sequence Diagrams

Users of the prototype are not restricted to these visualizations. External applications can be executed by every abstraction, hence data can be emitted to and visualized by external scripts, for example to draw a movement track in gnuplot. A graphical user interface implemented with HTML 5 and WebGL is provided as well.

The influence of data quality on different analyses is assessed in section 9.5 in regard to tolerance to errors in measurement and the availability of measured values of the z-axis. In order too cope with possible data quality problems, alternative approaches for some events were sketched where this might be necessary.

The presented techniques and notations are not restricted to bats. Models for activity transitions as well as all presented analyses are applicable to a wide range of species. The notation for continuous queries and event patterns only include some parts which were specifically added for the scenario: functions and operators for terrain features. However, region definitions are useful for many scenarios and their notation is certainly not restricted to BATS.

In summary, all objectives which were presented in section 1.3 are achieved.

9.2 Flexibility

Notations were presented to specify activities, events and terrain features. All important characteristics of behavior are user-defined. Hence, the only difficulty to use another model describing activities is creating the model and implementing events which were stated in the model as a transition.

If the model was adapted for another species, all already implemented analyses can be reused. Heatmaps and local convex hulls are useful analyses for a wide range of species. Moreover, predictions of behavior may only need minor changes in the suspected influence relations of available emissions. Inference on behavior is also available as soon as data is gathered and CPTs are sufficiently filled.

Additionally, changing the source of position data is not a problem. For example, the prototype could use position data gathered from GPS sensors attached to larger animals or from mobile phones. Events only need to know x, y and possibly z values of positions as well as expected precision.

In summary, the presented techniques are flexible in regard to other models and event definitions for different species as well as the source of examined position data.

9.3 Predictions

In order to separately evaluate the techniques to detect and predict activities, a simulator emitting only activities and their attributes has been elaborated. This allows to precisely restrict randomness for each activity and attribute.

9.3.1 Activity Simulator

Figure 9.1 shows the basic approach to generate activity sequences which are used to evaluate predictions. A user-defined timespan restricts the simulation. At first, the simulator checks whether the bat is sleeping. If it is sleeping, then the wakeup time and next sleeping interval has to be determined. If it is not sleeping, but the next sleeping interval already started, then `Sleeping` is the next activity. If the last activity has not been `Foraging`, then it is randomly assigned according to table 9.1. If `Foraging` was not randomly assigned or if it was the last activity, then `Pause` or `Drinking` is randomly assigned according to table 9.2.

#ateInsect	< 7	7 - 10	≥ 10
P(Foraging)	0.8	0.7	0.2

Table 9.1: Roll a Dice: Foraging

#watered	0	1	≥ 2
P(QuenchingThirst)	0.6	0.3	0.05
P(Pause)	0.4	0.7	0.95

Table 9.2: Roll a Dice: QuenchingThirst / Pause

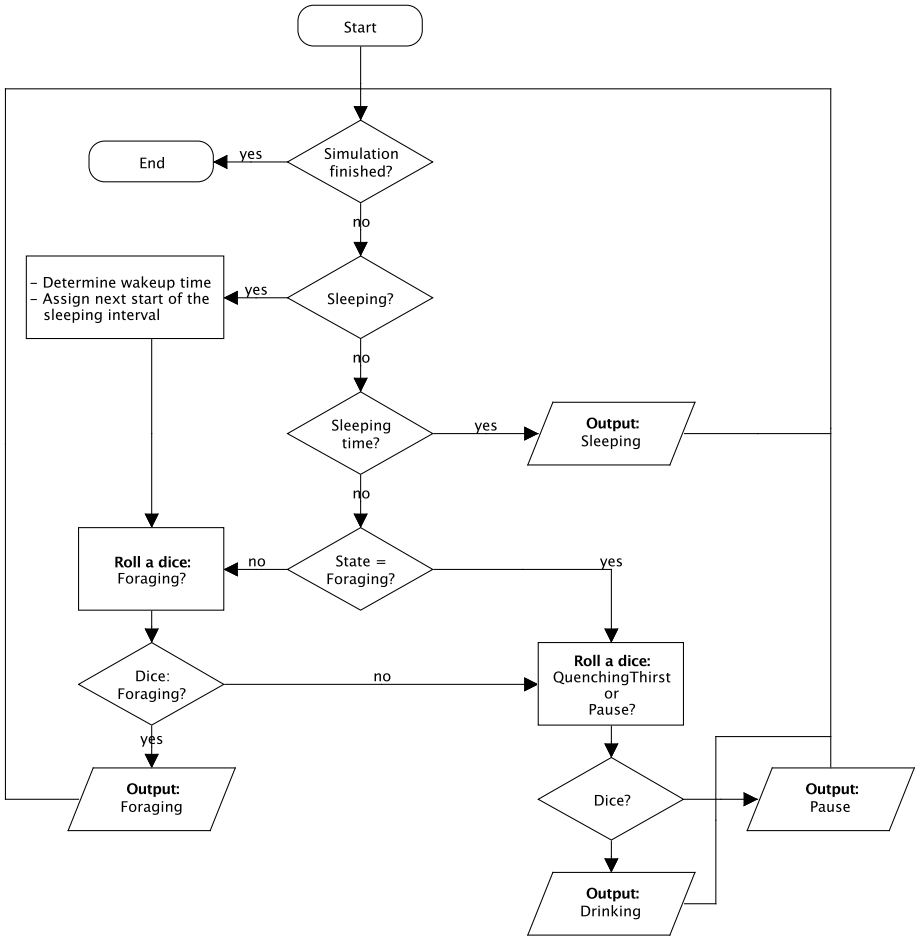


Figure 9.1: Simulator of Activity Sequences

9.3.2 Available Data

The simulated behavior is not affected by many influences. Table 9.1 shows that already eaten insects are important. In table 9.2 the previous visits at watering ponds affect probabilities. Since both values change during a night, time is considered as well. Figure 9.1 clearly shows, that the previous activity restricts possible following activities. Hence, four random variables are used for predictions: Time, Before, Ate, Drank.

Drank and Ate are both divided into *nothing*, *some* and *a lot*. For drinking, *some* means one previous drinking activity, whereas all other values are *nothing* respectively *a lot*. Ate is also aggregated by counting previous activities: *some* means one or two previous foraging activities. Time is partitioned to *daytime*, *dusk*, $night_1$, $night_2$ and *dawn*.

9.3.3 Results

All four available random variables have been used in different CPTs. The case of no available prediction was considered with a backup CPT which has no conditions. Other used CPTs are hypotheses on possible influence. *Adaptive Prediction* uses those six hypotheses, the backup CPT and the success rates of all CPTs to always emit the prediction of the previously most successful CPT.

Table 9.3 shows all hypotheses which are listed in the left column. The success is measured in 50 test runs for each configuration: 1 week, 1 month and 3 month with and without learning samples of 20 bats for 14 days. The observed timespan of learning data was chosen since it is assumed that bats are going to carry the mobile nodes for at least this timespan. The sample size of 20 bats also is in the range of observed bats in a possible initial deployment. Adaptive prediction assumes that a CPT is reliable enough, if two previous occurences of the current situation were observed.

Type	1 w -	1 mo -	3 mo -	1 w 20/14d	1 mo 20/14d	3 mo 20/14d
- (Backup)	13.73%	35.54%	40.02%	41.56%	42.06%	42.02%
Ate	29.05%	37.33%	43.81%	45.55%	46.68%	46.77%
Time, Before	36.92%	60.83%	66.82%	71.14%	71.82%	71.08%
Time, Ate	29.47%	45.13%	49.62%	52.32%	52.45%	51.88%
Ate, Before	42.51%	58.79%	63.81%	65.52%	66.88%	66.71%
Time, Ate, Before	29.17%	56.44%	64.98%	71.00%	71.46%	71.03%
Time, Ate, Before, Drank	17.00%	49.25%	62.79%	72.95%	73.85%	73.69%
Adaptive Prediction	48.90%	65.27%	70.09%	73.15%	74.03%	73.91%

Table 9.3: 50 Test Runs: Advantage of Adaptive Prediction Over Time

Table 9.3 clearly shows that over time all CPTs gain precision. Moreover, at least for the test runs, adaptive prediction improved the precision of predictions. At the beginning, adaptive prediction improved the emitted predictions significantly. Over time, it will only use the single most successful CPT and therefore their success will converge. Since initial predictions are more precise, it still yields a better success rate than the single most successful CPT.

Table 9.4 shows the overall success of predictions for different timespans. For no learning data and one observed bat, 50 test runs were executed for each timespan. The average success rate as well as the best and worst single test run are depicted. The predictions quickly improve over time until they can be considered useful after one week. The success rates should be interpreted considering the amount of randomness in table 9.1

Timespan	Average	Best	Worst
1 day	**18.16%**	30.77%	0.00%
2 days	**26.04%**	36.36%	8.33%
1 week	**49.39%**	57.35%	39.34%
2 weeks	**59.26%**	69.06%	52.52%
1 month	**65.84%**	69.93%	60.15%
2 months	**68.47%**	72.70%	64.12%
3 months	**70.09%**	73.61%	67.38%
6 months	**72.17%**	74.66%	69.27%
1 year	**72.71%**	73.75%	71.10%

Table 9.4: 50 Test Runs: 1 Bat, No Previous Data

and table 9.2. If actual bat behavior is less random, the approach most likely yields even better predictions.

Timespan	Average	Best	Worst
1 day	**81.07%**	100.00%	50.00%
2 days	**76.48%**	94.12%	56.52%
1 week	**74.98%**	83.33%	64.18%
2 weeks	**73.71%**	80.62%	61.07%
1 month	**74.03%**	77.89%	68.51%
2 months	**73.67%**	77.45%	70.31%
3 months	**73.62%**	76.35%	69.93%
6 months	**73.71%**	75.92%	70.64%
1 year	**73.73%**	75.04%	72.04%

Table 9.5: 50 Test Runs: 1 Bat, Previous Data: 20 Bats Over 2 Weeks

Table 9.5 shows the overall success of predictions if previous learning data is available. Obviously, the results are useful from the beginning.

In summary, predictions in the field are not going to add value in the first few days. After a short period of gathering enough learning data, predictions will be useful from the beginning of each observation. The precision of predictions depends on the hypotheses or rather the implemented CPTs as well as the randomness of bat behavior.

9.4 Detecting Events

In order to examine how well events can be expressed in the prototype, the notation of high level events has been examined in regard to which temporal relationships it can reproduce. Moreover, it was assessed whether the prototype is able to detect events on a synthetic position data stream.

9.4.1 Temporal Relationships Between Events

In order to prove that the notation for high level events presented in section 5.3 is expressive enough for all possible temporal relations between two events, patterns are provided for each relation. Table 9.6 is an overview of all thirteen possible relations listed by [All83].

Relation	Symbol	Symbol for Inverse	Pictoral Example
X before Y	$<$	$>$	`XXX YYY`
X equal Y	$=$	$=$	`XXX` `YYY`
X meets Y	m	mi	`XXX` `   YYY`
X overlaps Y	o	oi	`XXX` ` YYY`
X during Y	d	di	` XXX` `YYYYY`
X starts Y	s	si	`XXX` `YYYYY`
X finishes Y	f	fi	`  XXX` `YYYYY`

Table 9.6: The Thirteen Possible Relationships [All83]

The following list contains expressions for each relation:

- **X before Y:**
  $x = X \text{ -}>_{(\infty)} Y(end < y.start)$
- **X equal Y:**
  $x = X \mid\mid Y(start = x.start \wedge end = x.end)$

- **X meets Y:**
  X ->$_{(0s)}$ Y
- **X overlaps Y:**
  $X \mid\mid Y$
- **X during Y:**
  $x = X \mid\mid Y(x.start \geq start \wedge x.end \leq end)$
- **X starts Y:**
  $x = X \mid\mid Y(start = x.start)$
- **X finishes Y:**
  $x = X \mid\mid Y(end = x.end)$

Hence, the notation of high level event patterns allows to express all possible temporal relations.

9.4.2 Expressing Events

The sensor network is not yet deployed and there are still many assumptions on the behavior of sensors as well as the behavior of bats. Hence, event detection is tested on synthetic position data streams.

This evaluation does not check the ability to express *all* events which need to be detected in the field, since it is highly probable that event definitions will change according to actual gathered position data. Instead, *one* exemplary event is evaluated. Other events can be examined in a similar way.

However, it is more sensible to use the simulator in order to check how different assumptions would impact the behavior and possible patterns. A full evaluation of detectability of events and suitability of their notation for the scenario should be elaborated on actual data collected in the deployed sensor network.

The event which is evaluated is $ateInsect$. This section assumes that no z-axis is available or sufficiently reliable. It is expected that after successfully catching an insect a bat flies in circles for at least $60s$ while eating its prey. Moreover, it grubs for at least $22s$ until it finds a suspected beetle. The simulation circles for around $80s$ and grubs $25s$ for the insect. The pattern $pause_{22s} \triangleright flewInCircles_{60s}$ was used to detect $ateInsect$. The following streams were implemented according to section 5.6:

1. `SubsequentDistances`
2. `TotalDistance22s` (for $paused_{22s}$)

3. `TotalDistance5s` (for $paused_{22s}$)
4. `paused22s`
5. Steps 1 to 5 of `flewInCircles`

The streams `paused22s` and `flewInCircles` were interpreted as low level events. Listing 9.1 depicts all pattern definitions, which also provide duplicate elimination.

```
1 create event
2 ateInsectDummy(a.ID) ::= a=paused22s ->(10s) flewInCircles;
3
4 create event
5 ateInsectDuplicate(a.ID) ::= a=ateInsect || *ateInsectDummy;
6
7 create event
8 ateInsect(a.ID) ::= *ateInsectDummy;
```

Listing 9.1: Definition of *ateInsect*

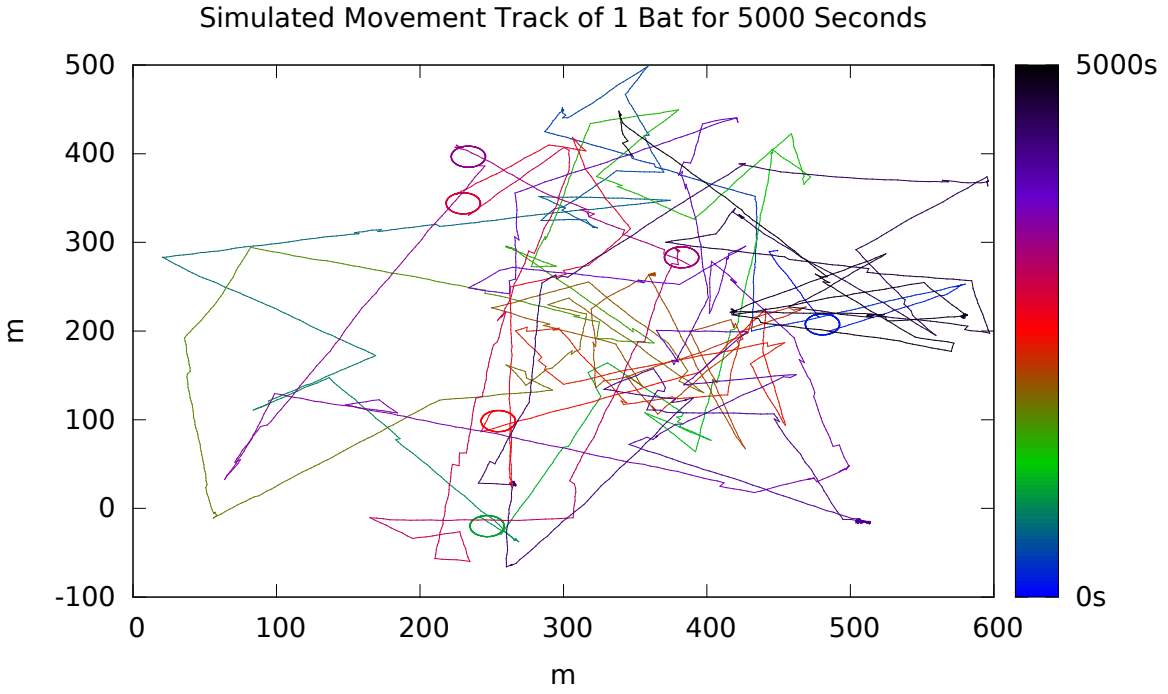


Figure 9.2: Simulated Movement Track

Figure 9.2 shows the simulated movement track which is examined. In order to also incorporate time, the color of a line changes during the hunting session. Ellipses represent

the assumed circular flights. In order to doublecheck, the simulator had to emit its internal state additionally to position data: six successful occurences of eating an insect were simulated.

Table 9.7 shows an excerpt of detected events which were defined in listing 9.1 and table 9.8 shows all emitted *ateInsect* events.

Type	Start	End
ateInsectDummy	68	150
ateInsect	68	150
ateInsectDummy	69	151
ateInsectDuplicate	68	151
ateInsectDummy	70	152
ateInsectDuplicate	68	152
...	...	
ateInsectDummy	903	985
ateInsect	903	985
ateInsectDummy	904	986
ateInsectDuplicate	903	986
...	...	

Table 9.7: Excerpt of Events (Listing 9.1)

Type	Start	End
ateInsect	68	150
ateInsect	903	985
ateInsect	2480	2562
ateInsect	2747	2829
ateInsect	3133	3215
ateInsect	3293	3375

Table 9.8: Detected *AteInsect* Events

In summary, all simulated *ateInsect* events were detected and there were no false positives. If the assumptions on the behavior are in accordance with reality, then no z-axis is necessary. Moreover, at least *ateInsect* is expressable in the prototype.

9.5 Influence of Data Quality on Analyses

Data quality influences several abstractions of analyses. Hence, this influence is discussed separately for each abstraction level.

9.5.1 Low Level Events

The influence of uncertainties and the availability of the z-axis have varying impact on low level events. Hence, all events which were presented in previous chapters are examined:

$\boldsymbol{Paused_{timespan}}$ and $\boldsymbol{Flew_{timespan}}$

The distinction of $paused_{timespan}$ and $flew_{timespan}$ does not require exact positions. However, measured positions should not continuously bounce. If expected random jumps of measurements within a radius are larger than regular flights, then $paused_{timespan}$ can no longer be detected. $Flew_{timespan}$ tolerates larger errors in measurement and continuous bounces if only distances have to be detected which are significantly longer than expected errors.

$\boldsymbol{FlewInCircles}$

The event $FlewInCircles$ is already specified without expecting circular trajectories at all. The definition assumes that during a defined timespan a total distance is exceeded and most of the measured points are in a defined radius. The event is very tolerant to errors in measurements, if the radius is larger than expected errors and a reasonable minimum for the total distance is applied.

$\boldsymbol{RisenUp}$ and $\boldsymbol{WentDown}$

Both events obviously depend on the existence and reliability of the z-axis. For the purpose of detecting $ateInsect$, the measured z-axis should at least indicate whether an animal is above the ground. Otherwise, both events are useless and should not be specified at all.

$\boldsymbol{EnteredRegion(X)}$ and $\boldsymbol{LeftRegion(X)}$

If soft discretization is applied and the regions to be detected are sufficiently large, then the events $enteredRegion(X)$ and $leftRegion(X)$ are very tolerant to errors in measurement. The border has to be specified according to expected errors. After that, jumps within an anticipated radius are no problem at all.

$\boldsymbol{FlewInSpeedCorridor_{range,timespan}}$, *accelerated*, *decelerated*

The tolerance of $flewInSpeedCorridor_{range,timespan}$ to errors in measurement depends on the characteristics of errors. If errors shift movement tracks to another location, then speed roughly remains the same. This shifting may be caused by applying the Kalman filter to positions. In this case, $flewInSpeedCorridor_{range,timespan}$ should be tolerant to measurement errors. If random jumps are expected, then there are several strategies, for example applying the Kalman filter. Resampling to fewer points may also decrease the effects of jumps. If this does not sufficiently help to approximate the speed, then the range should be extended. In summary, $flewInspeedCorridor_{range,timespan}$ should be tolerant to measurement errors, if the correct strategy for actual error characteristics in the field is applied.

The events *accelerated* and *decelerated* also depend on a good approximation of speed. The strategy to enable $flewInSpeedCorridor_{range,timespan}$ can also be applied to those events. Since the observed timespans are shorter, low data quality has higher influence on the detection.

changedDirection

The observed timespan for one segment should depend on the errors in measurement. *ChangedDirection* should be reliably detectable if the length of a segment is significantly longer than errors in measurement. At foraging speed, a bat may fly 4 to 10m in a second. If *changedDirection* has to be detected within a timespan of 15 seconds, the length of each half of the trajectory should be at least 30m. Hence, data quality will definitely influence the angle of changes. Generally changing the direction should nonetheless be detectable even for deviations of tens of meters.

$\boldsymbol{met(X, Y)}$

The event $met(X, Y)$ can rely on both position data and the meeting stream. Hence, it can use the alternative which shows higher precision in the field. The less precise stream may be used as a sanity check.

9.5.2 High Level Events

The influence of data quality on high level events depends on their combination of low level events. Hence, the events *paused* and $flew$ as well as $flewInSpeedCorridor_{range}$

are pretty tolerant to errors in measurement. The events *slowHuntingFlight* and *highTransitFlight* depend on $flewInSpeedCorridor_{range}$ and are therefore tolerant to errors as well.

The most problematic high level event is *ateInsect*. It directly depends on the availability and reliability of the z-axis, since it contains *risenUp* and *wentDown*. If the z-axis is definitely not available or sufficiently reliable, then the behavior can be approximated with a less accurate pattern:

$$pause_{\delta} \triangleright flewInCircles.$$

The parameter δ has to be estimated by biologists and should anticipate errors in measurement. Of course, whether this pattern suffices for actual bat behavior has to be evaluated on real data.

9.5.3 Activities and Predictions

Events are used as transitions between activities. There are hard to distinguish activities, which are grouped to *flying* and *resting*. The *resting* activities resembling sleeping and pausing only rely on whether *paused* can be detected. They are distinguished by the duration of a break. Errors in measurement might influence the detection of the duration in the range of several seconds to a minute. Since the sleeping activity takes several hours, the effects of data quality on distinguishing *sleeping* and *resting* is insignificant if *paused* can be detected at all.

The three *flying* activities are harder to separate. If the event *ateInsect* is not available, characteristics of the movement track have to be used to detect *hunting*, e.g. a rather low speed and frequent changes of the animals direction. Transit flights are faster and the direction rarely changes. Hence, *transit* and *hunting* depend on a reliable detection of $flewInSpeedCorridor_{range}$. It may be necessary to change specified speed corridors if errors in measurement are worse then expected. Distinguishing *hunting* and *drinking* depends on the specified regions for water ponds. If regions for water ponds are larger than expected errors, *drinking* can reliably be detected. However, the data quality of region definitions is significant.

The definitions of activities do not depend on the availability of the z-axis. Overall, they are tolerant to errors in measurement.

Values of observed random variables, e.g. temperature and humidity, are partitioned into a small amount of intervals to keep CPTs manageable. Rough classifications do not rely on precision, hence observed random variables and thereby CPTs are insignificantly

influenced by measurement errors. Since predictions only depend on the reliable detection of random variables and activities, they do not depend on the availability of the z-axis and are tolerant to errors in measurement, too.

9.5.4 Visualization

Heatmaps and Local Convex Hulls are state-of-the-art methods to visualize position data in larger areas. Especially regular rastered heatmaps are preferably coarse-grained and therefore tolerant to errors in measurement. Given that regions in choropleth maps are sufficiently large, this holds for choropleth maps, too. Local convex hulls are, similar to heatmaps, also detected for larger observation areas. Deviations in the range of meters to tens of meters may influence the position of boundaries in local convex hulls. Nonetheless, the shape of LoCoHs should roughly remain.

9.6 Summary

This chapter starts with a discussion on whether all objectives were achieved and affirms this question. Afterwards, the flexibility of the prototype in regard to other species and different data sources is explained.

In order to evaluate predictions, a simulator only emitting activities and their attributes is elaborated. With this simulator, results of several CPTs as well as Adaptive Prediction with different amounts of learning data and observation intervals are compared. Adaptive Prediction improves the precision of predictions.

The expressibility of temporal relationships between events was assessed. The proposed notation is able to express all thirteen possible relations listed by [All83]. Moreover, expressing events which are necessary for this scenario was exemplarily elaborated for $ateInsect$.

Finally, the influence of data quality on analyses was discussed. Most stated events are pretty tolerant to errors in measurement. The z-axis is not necessary. However, a reliable z-axis is still desirable in order to increase precision and to reduce false positives.

10 Conclusion

Chapter 1 introduced the objectives of this thesis. Biologists should be supported in answering questions on the behavior of bats. Moreover, mechanisms should be provided which trigger experiments and enable integration of external processing of data. Events and activities were chosen to describe animal behavior. Hence, the prototype provides notations and detects user-defined events and activities. Moreover, the prototype provides several state-of-the-art and new visualizations. The influence of data quality on analyses was assessed and proposed notations and techniques are applicable to a wide range of problems. All objectives are achieved.

This thesis was elaborated in the context of the DFG research group 1508 (BATS). The scenario is introduced in chapter 2. Furthermore, the chapter states biologists questions on the behavior of the greater mouse-eared bat.

Afterwards, projects related to BATS are presented. Aspects of the notation and implementation of related work are incoroporated into the presented approach, for example the integration of data stream processing and complex event processing as well as notations of terrain features.

Chapter 4 introduces the fundamentals to the presented approach. The model of PIPES was chosen for data stream processing in the prototype, since its operators are well-defined and the model emphasizes on application time. Esper and its Event Processing Language (EPL) had impact on the notation and approach to event detection. Both systems as well as techniques of machine learning are covered in this chapter.

The main aspect of this thesis are events and activities. Chapter 5 introduces the terms *event* and *activity* and provides several notations for different abstractions. Events are separated into *low level events* and *high level events*. Low level events are results emitted from regular continuous queries in the data stream processing model which are interpreted as events. High level events are patterns over low and high level events and therefore integrate complex event processing into the proposed approach. In order to model activities, UML state diagrams are extended with revision transitions and aliases. It has been proven that the proposed notation for event patterns is able to express all temporal relations between two events. Several examples of events and activities which

are important to the scenario have been exemplarily implemented in order to show that they can be expressed in the proposed notations.

The definition of activities enables to use Dynamic Bayesian networks to predict them. For the purpose of simplifying the notation, domain experts state Bayesian networks that consist of environmental emissions without any state transition and the transition of animal state. Suitable random variables to be used in Bayesian networks are presented. The approach to predict activities is also applied to attributes of activities. Adaptive prediction is proposed in order to improve the precision of predictions. In the evaluation chapter, adaptive prediction was compared to single CPTs and always yielded better results.

Heatmaps, LoCoHs and activity sequence diagrams are presented in chapter 7. Moreover, inference on Bayesian networks is introduced. Special cases of inference are expected activities and a proposed definition of personality of an animal.

Chapter 8 explains the implementation of the prototype in more detail. Available operators of data stream processing are described. Additional operators for terrain features are introduced. Available operators for event detection are listed afterwards. The prototype supports event consumption. Integrating predictions into the prototype as well as into data stream systems in general is explained. An existing simulation framework was extended with more realistic behavioral patterns in order to evaluate their detection. The evaluation in chapter 9 uses synthetic data of the altered simulation framework to show that the prototype is able to detect complex patterns like the consumption of an insect after a successful hunt.

Additionally to already summarized results of the evaluation chapter, the effects of data quality on the detection of events and activities as well as on results of predictions and visualizations was discussed. The z-axis is not necessary to detect events in this scenario. Most of the stated events are pretty tolerant to errors in measurement or can be expressed differently to anticipate lack of precision.

Of course, there are still topics which need to be tackled in the future. The notation and approach to activity detection assumed that at any time instant exactly one activity could be assigned. A notation incorporating nondeterminism could be used to cover exceptions to this assumption and to avoid retroactive revisions of already emitted activities.

If sensors produce several hypotheses on the position of a tracked object, all hypotheses could be used in the proposed abstraction layers. This would significantly increase complexity, but could improve precision of event and activity detection as well as

predictions and analyses. Predictions of events would allow to further support experiments and they could improve predictions of activities. Conditional tables could also be enhanced to support continuous functions, which could improve inference.

Moreover, in this scenario positions are measured by one base station and passed to the prototype. If mobile nodes could measure their own position, which for example is the case by using GPS sensors, then distributed processing and distributed execution plans deserve a closer look. If mobile nodes process their position data to events and activities themselves, then application time depends on the precision of local clocks. The detection of events concerning different tracked objects could use vector clocks to anticipate effects of divergent clocks.

Finally, the presented notations and techniques should be applied to other problems.

Glossary

Activity

Every observed object performs some activity at each time instant. Two activities cannot be valid for the same object at the same time. A notification of an activity has to include the related tracked object. A sequence of activities resembles the behavior of an animal.

Activity Sequence Diagram

Activity sequence diagrams visualize sequences of activities and predictions. The axes of activity sequence diagrams are a time line as well as distinct values (*lanes*) for the IDs of tracked animals. Events may be blended into the lanes of the diagram.

Adaptive Prediction

Using the previously most successful conditional table to predict the next activity.

Alias

State which has its own in- and outgoing transitions, but is visible to the user as another state. Useful for dummy states and more complex networks and rules of transitions.

Annotation

An annotation is any additional data that is attached to an event.

Baum-Welch Algorithm

The Baum-Welch algorithm is a well-known iterative approach for adjusting model parameters in a ↑Hidden Markov Model [RJ86].

Bayesian Network

A Bayesian network is a directed acyclic graph in which every node is labeled with information on its probability distribution [RN04].

Broker Operator

The broker operator enables time-coordinated processing of cyclic queries [Bol11].

Central Station

The central station in BATS is a common desktop or laptop computer communicating with the ↑Stationary Sensor Nodes. It stores gathered data and executes the presented methods and techniques to analyze and process position and environmental data streams.

Choropleth Map

Choropleth maps are common approach to map data to regions of arbitrary shape. Based on some metric, regions are colored following a color scale [Yau11]. Choropleth maps are a special case of ↑Heatmaps.

Complex Event Processing

"The complex event processing model views flowing information items as notifications of ↑Events happening in the external world, which have to be filtered and combined to understand what is happening in terms of higher-level events." [CM12]

Concatenation

A concatenation is a ↑Pattern of immediately following ↑Events. In the context of ↑Spatio-temporal Predicates, the concatenation $P \triangleright Q$ combines two immediately following spatio-temporal predicates P and Q [BBBB10].

Conditional Average Table

Conditional average tables are similar to ↑Conditional Probability Tables, but display averages instead of probabilities. Continuously updating a conditional average table is trivial. Conditional averages can directly be emitted as possible lengths of activities or for other attributes.

Conditional Histogram Table

Conditional histogram tables are similar to ↑Conditional Average Tables, but display histograms instead of averages. Continuously updating a conditional histogram table is trivial. Conditional histogram tables are intended to be interpreted by domain experts.

Conditional Median Table

Conditional median tables are similar to ↑Conditional Average Tables, but display the median instead of an average. The median requires a full history of previous values. A conditional median table is also only intended for results visible to domain experts.

Conditional Probability Table

A conditional probability table (CPT) represents a conditional probability distribution of discrete random variables [RN04].

Conditional Region Table

Conditional region tables are ↑Conditional Probability Tables which display the probability of regions or sets of regions.

Contextual Data Repository

A contextual data repository is an external source of knowledge that provides contextual data, e.g. a database of terrain features. It is a source for ↑Annotations.

Continuous Query

Continuous Queries "run continuoulsy over a period of time and incrementally return new results as new data arrives" [GÖ03].

Data Stream Management System

A Data Stream Management System (DSMS) manages schema definitions of incoming ↑Streams as well as ↑Continuous Queries and offers access protection [MW13].

Data Stream Processing

In the data stream processing model, ↑Streams of data coming from different sources are processed to produce new data streams as an output [CM12].

Data Stream System

A Data Stream System (DSS) consists of a ↑Data Stream Management System, data stream definitions and stored ↑Continuous Queries [MW13].

Dynamic Bayesian Network

Dynamic Bayesian Networks (DBNs) extend ↑Bayesian networks with time and allow more compact representations of states and ↑Conditional Probability Tables than ↑Hidden Markov Models.

Emission

Emissions are observable sensor data. They are observed with a certain probability depending on the actual state.

Event

An event is a message that something happened at a specific point of time or timespan. If it belongs to a tracked object, it can and should be attributed to it. Several events attributed to the same object are allowed to happen at the same time. Events do not need to happen at every time instant.

Event Processing Language

The Event Processing Language (EPL) is a rich declarative language for rule specification which is part of Esper [CM12].

Forward-Backward Procedure

The forward-backward procedure finds the probability of a given observation sequence in a ↑Hidden Markov Model [RJ86].

Hard Discretization

For ↑Regions, hard discretization applies a region test and does not consider previous results. Moving around the border of a region yields many events of entering and leaving the region. ↑Soft Discretization reduces the amount of messages.

Heatmap

In a heatmap, values are represented by colors. There are several types of heatmaps. A heat matrix is a typical grid with colored cells. ↑Choropleth Maps are heatmaps using regions of arbitrary shape instead of rectangular cells of a table. The prototype optionally overlays heatmaps over satellite images to support domain experts in relating values to ↑Terrain Features regardless of whether they were modeled as ↑Regions.

Hidden Markov Model

A Hidden Markov Model is a doubly stochastic process that models both actual ↑States and observable ↑Emissions [RJ86].

High Level Event

A high level event is a ↑Pattern over ↑Low Level Events or other high level events.

Hole

A hole is an interval of data accidentally missing due to hardware or software malfunction [PSR+13]. Not to be confused with a ↑Semantic Gap.

Home Range

A home range can be defined as "the normal area used by an animal in its life activities" [ADMW09].

Kalman Filter

The Kalman filter uses a series over several (imprecise) state vectors to generate an estimation of the actual state vector [Bol11]. The approach is similar to ↑Dynamic Bayesian networks.

Local Convex Hull

Local convex hulls estimate ↑Home Ranges and utilization distributions. They are able to identify hard boundaries and are able to display holes in the distribution [GFRC+07].

Low Level Event

Low level events are regular continuous ↑Streams interpreted as ↑Events.

Lunaphobia

Lunaphobia is the fear of the moon or rather moonlight. It does not have to be fear of the moon itself, instead animals may be more aware if predators are more likely to find them and therefore show signs of lunaphobia.

Mobile Sensor Node

Mobile sensor nodes are attached to bats. They emit signals in order to enable ↑Stationary Sensor Nodes to measure their position.

Movement Track

A movement track is a sequence of spatio-temporal positions, i.e. a sequence of (instant, point) pairs, of a tracked object [PSR+13].

Operator

An operator is a atomic processing unit in a ↑Data Stream Management System. It has incoming and outgoing ports. An operator processes incoming data continuously and emits results to an outgoing port [Bol11].

Pattern

A pattern is the definition of a ↑High Level Event which is comprised of several other events. It states temporal relations, i.e. sequential, parallel and alternative processes, as well as temporal conditions.

Projection

The projection ↑Operator applies a mapping function to incoming data tuples in order to emit altered ones.

Raw Trajectory

"A raw trajectory is a trajectory extracted from a raw movement track and containing only raw data for its Begin-End interval." [PSR+13]

Region

A region is a user-defined area of interest. Rectangles, circles and polygons are available for its definition. Moreover, a region can consist of two regions R_1 and R_2 combined to $R_1 \cup R_2$, $R_1 \cap R_2$ or $R_1 \setminus R_2$.

Revision Transition

In activity detection, revision transitions replace the last chosen activity with a revised one. Not to be confused with a regular ↑Transition.

Selection

The selection ↑Operator filters incoming data tuples in regard to whether they fulfill a given predicate. Tuples not fulfilling the predicate are discarded.

Semantic Enrichment

Semantic enrichment is the process of adding knowledge to a trajectory [PSR+13].

Semantic Gap

A period of data missing on purpose is called semantic gap. Not to be confused with a ↑Hole.

Semantic Trajectory

"A semantic trajectory is a trajectory that has been enhanced with ↑Annotations and/or one or several complimentary segmentations." [PSR+13]

Snapshot-Reducibility

Snaphot-reducibility is the characteristic of a data stream processing ↑Operator, that it resembles the behavior of its relational, non-temporal counterpart for each ↑Time Instant [Krä07a].

Soft Discretization

For ↑Regions, soft discretization applies a region test and considers previous results. It adds a tube around the border in which it is unclear whether a point is inside a region. Within the tube, the previous state remains unchanged. Hence, soft discretizations returns less events than ↑Hard Discretization if a monitored animal is moving around the border of a region.

Spatio-temporal Predicate

"A spatio-temporal predicate $P(x, R)$ is a function that returns T, F or $\perp$, depending on the topological relation of x to R over time." [BBBB10] Spatio-temporal predicates correspond to ↑Low Level Events.

State

The state contains present conditions of a system or a single object. In ↑Hidden Markov Models, the state is resembled by one discrete random variable and can only be implicitly observed by ↑Emissions.

Stationary Sensor Node

A network of ground-based stationary sensor nodes combines measurements of emitted signals from ↑Mobile Sensor Nodes to position hypotheses. Stationary sensor nodes are an interface to both mobile sensor nodes and the ↑Central Station. Additional sensors can be attached in order to measure environmental data.

Stream

"A data stream is a real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamp) sequence of items." [GÖ03]

Terrain Features

Terrain features are spatial characteristics of the environment, e.g special areas like ponds or rivers as well as areas in which a certain type of tree grows. Terrain features are modeled as ↑Region.

Time Instant

A time instant is the smallest possible time interval.

Transition

In activity detection, low and high level events can act as transitions between two activities.

Viterbi Algorithm

"The Viterbi algorithm is a recursive optimal solution to the problem of estimating the state sequence of a discrete finite-state Markov process observed in memoryless noise." [FJ73]

Window

The window ↑Operator allows to combine tuples of several subsequent ↑Time Instants. The implementation in the prototype corresponds to time-based sliding windows ω^{time} which "shifts a time interval of size w time units over its input stream to define the output stream" [Krä07a].

Bibliography

[ADMW09] Sybill Amelon, David C Dalton, Joshua J Millspaugh, and Sandy A Wolf. Radiotelemetry; Techniques and Analysis. *Ecological and Behavioral Methods for the Study of Bats/edited by Thomas H. Kunz and Stuart Parsons*, 2009.

[AFR+10] Darko Anicic, Paul Fodor, Sebastian Rudolph, Roland Stühmer, Nenad Stojanovic, and Rudi Studer. A Rule-Based Language for Complex Event Processing and Reasoning. In *Web Reasoning and Rule Systems*, pages 42–57. Springer, 2010. URL: `http://elibrary.palcomtech.ac.id/wp-content/uploads/Web-Reasoning.pdf#page=52` [cited 2014-09-01].

[All83] James F Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983. URL: `http://web.cacs.louisiana.edu/~logan/521_f08/Doc/p832-allen.pdf` [cited 2014-09-01].

[BBBB10] Markus Bestehorn, Klemens Böhm, Patrick Bradley, and Erik Buchmann. Deriving Spatio-temporal Query Results in Sensor Networks. In M. Gertz and B. Ludäscher, editors, *Proc. SSDBM*, number 6187 in LNCS, page 6–23, Berlin Heidelberg, 2010. Springer-Verlag. URL: `http://link.springer.com/chapter/10.1007/978-3-642-13818-8_3` [cited 2014-09-01].

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. URL: `http://arxiv.org/pdf/0803.0476.pdf` [cited 2014-09-01].

[BLBGW10] Jó Ágila Bitsch Link, Thomas Bretgeld, André Goliath, and Klaus Wehrle. RatMote - A Sensor Platform for Animal Habitat Monitoring.

In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 432–433. ACM, 2010. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.164.3182&rep=rep1&type=pdf [cited 2014-09-01].

[Bol11] André Bolles. *Ein datenstrombasiertes Framework zur Objektverfolgung am Beispiel von Fahrerassistenzsystemen*. PhD thesis, Carl-von-Ossietzky-Universität Oldenburg, 2011.

[CH92] Gregory F Cooper and Edward Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine learning*, 9(4):309–347, 1992. URL: http://www.cs.ru.nl/~peterl/BN/cooper-hers.pdf [cited 2014-09-01].

[CJ09] Sharma Chakravarthy and Qingchun Jiang. *Stream Data Processing: A Quality of Service Perspective: Modeling, Scheduling, Load Shedding, and Complex Event Processing*, volume 36. Springer, 2009.

[CM12] Gianpaolo Cugola and Alessandro Margara. Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Computing Surveys (CSUR)*, 44(3):15, 2012. URL: http://www.researchgate.net/publication/220796090_Processing_flows_of_information_from_data_stream_to_complex_event_processing/file/9fcfd50cae6c0a512d.pdf [cited 2014-09-01].

[DEM+12] Vladimir Dyo, Stephen A Ellwood, David W Macdonald, Andrew Markham, Niki Trigoni, Ricklef Wohlers, Cecilia Mascolo, Bence Pásztor, Salvatore Scellato, and Kharsim Yousef. WILDSENSING: Design and Deployment of a Sustainable Sensor Network for Wildlife Monitoring. *ACM Transactions on Sensor Networks (TOSN)*, 8(4):29, 2012. URL: http://dl.acm.org/citation.cfm?id=2240118 [cited 2014-09-01].

[Deu] Deutscher Verband für Landschaftspflege. Großes Mausohr. URL: http://www.bayerns-ureinwohner.de/bayerns-ureinwohner/arten-steckbriefe/detailansicht/id/grosses-mausohr.html [cited 2014-09-01].

[DFG13] DFG Research Group 1508 BATS. Goals, 9 2013. URL: http://www.for-bats.org/goals.shtml [cited 2014-09-01].

[Doe93] Dan Doernberg. Computer Literacy Interview with Donald Knuth, 1993.

[Espa] EsperTech Inc. Esper - Complex Event Processing. URL: http://esper.codehaus.org [cited 2014-09-01].

[Espb] EsperTech Inc. Esper Reference. URL: http://esper.codehaus.org/esper-5.0.0/doc/reference/en-US/pdf/esper_reference.pdf [cited 2014-09-01].

[FJ73] G David Forney Jr. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973. URL: http://port70.net/~nsz/articles/classic/forney_viterbi_1973.pdf [cited 2014-09-01].

[GCP+06] Ying Guo, Peter Corke, Geoff Poulton, Tim Wark, Greg Bishop-Hurley, and Dave Swain. Animal Behaviour Understanding using Wireless Sensor Networks. In *Proc. 31st IEEE Conf. on Local Computer Networks (LCN, Tampa, Florida, U.S.A.)*, pages 607–614, 2006. URL: http://eprints.qut.edu.au/33808/1/33808.pdf [cited 2014-09-01], doi:10.1109/LCN.2006.322023.

[GFR13] Wayne M Getz and Scott Fortmann-Roe. LoCoH: Powerful Algorithms for Finding Home Ranges, 2013. URL: http://locoh.cnr.berkeley.edu [cited 2014-09-01].

[GFRC+07] Wayne M Getz, Scott Fortmann-Roe, Paul C Cross, Andrew J Lyons, Sadie J Ryan, and Christopher C Wilmers. LoCoH: Nonparameteric Kernel Methods for Constructing Home Ranges and Utilization Distributions. *PloS one*, 2(2):e207, 2007. URL: http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0000207 [cited 2014-09-01].

[GÖ03] Lukasz Golab and M Tamer Özsu. Issues in Data Stream Management. *ACM Sigmod Record*, 32(2):5–14, 2003. URL: http://eolo.cps.unizar.es/docencia/doctorado/Articulos/DataStreams/Issues%20in%20Data%20Stream%20Management.pdf [cited 2014-09-01].

[GW04] Wayne M Getz and Christopher C Wilmers. A local nearest-neighbor convex-hull construction of home ranges and utilization distributions. *Ecography*, 27(4):489–505, 2004. URL: http://www.jstor.org/stable/3683421?__redirected [cited 2014-09-01].

[HKP11] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining Concepts and Techniques*. Morgan Kaufman, 3 edition, 8 2011.

[JAC07] Qingchun Jiang, Raman Adaikkalavan, and Sharma Chakravarthy. MavEStream: Synergistic Integration of Stream and Event Processing. In *Proceedings of the Second International Conference on Digital Telecommunications*, page 29. IEEE Computer Society, 2007. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4270595 [cited 2014-09-01].

[Kal60] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960. URL: http://www.math.harvard.edu/archive/116_fall_03/handouts/Kalman1960.pdf.

[Khr14a] Khronos Group. Getting a WebGL Implementation, 2014. URL: http://www.khronos.org/webgl/wiki/Getting_a_WebGL_Implementation [cited 2014-09-01].

[Khr14b] Khronos Group. WebGL - OpenGL ES 2.0 for the Web, 2014. URL: http://www.khronos.org/webgl/ [cited 2014-09-01].

[Krä07a] Jürgen Krämer. *Continuous Queries over Data Streams-Semantics and Implementation.* PhD thesis, Universität Marburg, 2007.

[Krä07b] Jürgen Krämer. Continuous Queries over Data Streams-Semantics and Implementation. 2007. URL: http://www.dblab.ntua.gr/~gtsat/collection/data%20streams/443.pdf [cited 2014-09-01].

[LTG13] Andrew J Lyons, Wendy C Turner, and Wayne M Getz. Home Range Plus: A Space-Time Characterization of Movement Over Real Landscapes. *Movement Ecology*, 1(1):1–14, 2013. URL: http://tlocoh.r-forge.r-project.org/hrplus/lyons_turner_getz_2013_homerange-plus.pdf [cited 2014-09-01].

[LTM06] Olaf Landsiedel, Johannes Thiele, and Hanspeter Mallot. Rat Watch: Using Sensor Networks for Animal Observation. 2006. URL: `https://www.comsys.rwth-aachen.de/fileadmin/papers/2006/2006-06-Landsiedel-RatWatch.pdf` [cited 2014-09-01].

[Lyo14] Andy Lyons. T-LoCoH, 2014. URL: `http://tlocoh.r-forge.r-project.org` [cited 2014-09-01].

[MW13] Klaus Meyer-Wegener. Lecture: Datenstromsysteme. University of Erlangen-Nürnberg, 2013.

[NAB] NABU Schleswig-Holstein. Großes Mausohr. URL: `http://schleswig-holstein.nabu.de/naturvorort/fledermaeuse/fledermausarteninschleswig-holstein/02948.html` [cited 2014-09-01].

[NHK+14] Thorsten Nowak, Martin Hierold, Alexander Koelpin, Markus Hartmann, Hans-Martin Troger, and Jorn Thielecke. System and signal design for an energy-efficient multi-frequency localization system. In *Wireless Sensors and Sensor Networks (WiSNet), 2014 IEEE Topical Conference on*, pages 55–57, Jan 2014. `doi:10.1109/WiSNet.2014.6825497`.

[OTBW08] Okuary Osechas, Johannes Thiele, Jó Bitsch, and Klaus Wehrle. Ratpack: Wearable Sensor Networks for Animal Observation. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 538–541. IEEE, 2008. URL: `http://www.comsys.rwth-aachen.de/fileadmin/papers/2008/2008-08-Osechas-EMBC08-RatPack.pdf` [cited 2014-09-01].

[PSR+13] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, Jose Macedo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. Semantic trajectories modeling and analysis. *ACM Computing Surveys*, 45(4), August 2013. Article 42. URL: `http://dl.acm.org/citation.cfm?id=2501656` [cited 2014-09-01].

[RJ86] Lawrence Rabiner and Biing-Hwang Juang. An Introduction to Hidden Markov Models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986. URL:

http://www.cs.umb.edu/~rvetro/vetroBioComp/HMM/Rabiner1986%20An%20Introduction%20to%20Hidden%20Markov%20Models.pdf [cited 2014-09-01].

[RLH09] Bernd-Ulrich Rudolph, Alois Liegl, and Otto Von Helversen. Habitat Selection and Activity Patterns in the Greater Mouse-Eared Bat Myotis myotis. *Acta Chiropterologica*, 11(2):351–361, 2009. URL: http://wiki.for-bats.de/images/5/5e/Rudolph_et_al._2009.pdf [cited 2014-09-01].

[RN04] Stuart Russell and Peter Norvig. *Künstliche Intelligenz*, volume 2. Pearson Studium, 2004.

[RP87] George Rebane and Judea Pearl. The Recovery of Causal Poly-Trees from Statistical Data. *arXiv preprint arXiv:1304.2736*, 1987. URL: http://arxiv.org/pdf/1304.2736.pdf [cited 2014-09-01].

[RPÁTSA+13] José Ramón Rodríguez Pérez, María Flor Álvarez Taboada, Enoc Sanz Ablanedo, Antonio Gavela, et al. Comparison of GPS Receiver Accuracy and Precision in Forest Environments. Practical Recommendations Regarding Methods and Receiver Selection. 2013. URL: https://buleria.unileon.es/xmlui/bitstream/handle/10612/2709/Ramon.pdf?sequence=1 [cited 2014-09-01].

[Sta13] Statistisches Bundesamt. Deutsche Studierende im Ausland - Statistischer Überblick 2001-2011, 11 2013. URL: https://www.destatis.de/DE/Publikationen/Thematisch/BildungForschungKultur/Hochschulen/StudierendeAusland5217101137004.pdf [cited 2014-09-01].

[Ste13] Christian Steudtner. Konzeptionierung eines Simulations-Frameworks zur Erzeugung synthetischer Datenströme mit realistischer Qualität. 2013.

[TOBW08] Johannes Thiele, Okuary Osechas, Jó Bitsch, and Klaus Wehrle. Smart Sensors for Small Rodent Observation. In *Sensors, 2008 IEEE*, pages 709–711. IEEE, 2008. URL: https://www.comsys.rwth-aachen.de/fileadmin/papers/2008/2008-10-Thiele-Sensors08-RatPack.pdf [cited 2014-09-01].

[Yau11] Nathan Yau. *Visualize This: The FlowingData Guide to Design, Visualization, and Statistics*. John Wiley & Sons, 2011.

Curriculum Vitae

Friedrich-Alexander-Universität Erlangen-Nürnberg
Winter term 2012/2013 - Summer term 2014
Computer Science, M.Sc. (Minor: Business Studies)

Friedrich-Alexander-Universität Erlangen-Nürnberg
Winter term 2009/2010 - Summer term 2012
Computer Science, B.Sc. (Minor: Business Studies)

Maximilian-Kolbe-Fachoberschule Neumarkt (13th Grade)
September 2008 - July 2009
Abitur: Best final exams in Mathematics and Business Studies

Maximilian-Kolbe-Fachoberschule Neumarkt (11th + 12th Grade)
September 2006 - August 2008
Fachabitur: Best final exam in Mathematics
Internship in 11th grade: Neumarkt Tax Office

Knabenrealschule Neumarkt
September 2002 - August 2006

Willibald-Gluck-Gymnasium Neumarkt
September 2000 - August 2002

Grundschule Woffenbach-Neumarkt
September 1996 - August 2000

Scholarships

WS 2012/13 - WS 2013/14	Deutschlandstipendium (Siemens AG)
WS 2011/12 - SS 12	ATE-Stipendium (Alumni Technische Fakultät Erlangen e.V.)