State-based Context Prediction in Mobile Systems

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Stefan Föll

aus Tübingen

Hauptberichter:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Mitberichter:	Univ.-Prof. Mag. Dr. Alois Ferscha
Mitprüfer:	Prof. Dr. rer. nat. Volker Diekert
Tag der mündlichen Prüfung:	25. August 2014

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart
2014

Acknowledgments

I would like to express my gratitude to my supervisor Prof. Dr. Kurt Rothermel for his excellent support and guidance during my time at the University of Stuttgart. The big effort he puts into advising the work of his doctoral students has been a great contribution for this thesis. I am very grateful to have had the special opportunity to conduct research in his group, where one can find great conditions for learning and enhancing fundamental skills that are essential to academic research. Also, I would like to thank Prof. Dr. Alois Ferscha for accepting the role as a second reviewer. With the two theses of Dr. Rene Mayrhofer and Dr. Stephen Sigg, he already reviewed two contributions with great impact in the field of context prediction, and I feel very honoured that he is also a reviewer of my thesis.

Special thanks go to my project supervisor Dr. Klaus Hermann. I was always fascinated by his excellent communication and writing skills, and the constructive feedback he provided me with served as a great inspiration to improve my own ones.

At the University of Stuttgart, I had a great time during which I met brilliant researchers and - most importantly - warm people and colleagues. It was a humble experience to be part of a research group where colleagues helped each other not only in research, but in all aspects of academic life. While, as explored in this thesis, predictions are always uncertain to some degree, I am confident that there is no uncertainty involved when saying that I will stay in close contact with them in the future. Also, I would like to thank all of my colleagues whom I met in the European research project ALLOW. Having worked in an international research project over several years, I enjoyed very much collaborating with other researchers from universities all across Europe on interesting research problems.

Finally, I would like to thank my family and all my friends for their enduring support in every situation of my life. Only because of their strong encouragement, care and love, my academic endeavour has been made possible. I feel very happy to know that I can not only count on them during this important stage of my life, but also in the future whatever may happen next.

Contents

Abstract

Context-aware computing has developed from a pure research area to a widely acknowledged design principle of modern mobile systems over the last years. Mobile applications, able to automatically adapt to a user's dynamic context and improve the ease of human-computer interactions, are commonly available today. However, with current context-aware services such as restaurant finders or mobile tour guides, it is possible to support users with respect to their present behaviour only. As the next stage in context-aware computing more intelligent proactive applications are envisioned which can not only respond to the current, but also the future context of humans: Smart homes capable of controlling the ambient environment in expectation of the inhabitants' prospective actions; Social network applications which alert users about places where their friends might be going to; Personalized mobile recommender system to promote events and offers at venues which are relevant to the daily schedules of humans.

The development of suitable context prediction methodologies to turn such applications into a reality is however a challenge. The reason is that future context information, hidden in the raw context traces left by users in the real world, is not immediately accessible to applications. Therefore, sophisticated context prediction approaches are required that are able to discover and mine patterns of a user's behaviour from observed context histories. However, approaches which make accurate and expressive context predictions available and exploit this knowledge to optimize context-aware systems are missing in current research. As a consequence, the full potential of context-aware technologies has not been completely realised yet. In order to address this issue, we contribute in this work new context prediction algorithms and models for state-based context data, suitable for a range of different context types, such as a user's locations or activities. To this end, this thesis makes the following contributions.

In the first part of this thesis, we develop a novel context prediction system which applies statistical modeling concepts to automatically learn a machine-processable model of a user's behaviour and infer context predictions. With our context prediction system, we identify and address two shortcomings of existing approaches, prediction accuracy and prediction expressiveness, and propose suitable techniques and algorithms to improve

them. For increasing the prediction accuracy over current systems, we develop a new context predictor that is able to exploit the conditional dependency of context changes on a user's activities to anticipate forthcoming context states. Further, in order to overcome the limited expressiveness of prevailing prediction approaches, we explore the application of model checking algorithms for enabling expressive time-dependent forecasts in context prediction systems. Based on the algorithms and models developed in the first part of this thesis, we are able to significantly increase the amount and accuracy of the knowledge provided to proactive applications for the prediction of future context information.

In the second part of this thesis, we shift our attention towards tailored context prediction approaches to optimize the performance of mobile sensing applications. These applications represent a new class of mobile systems in the focus of current research, designed to forward streams of sensed context updates to interested parties over wireless communication channels. As mobile data communication induces a substantial energy overhead on mobile devices, we develop novel prediction-based protocols for improving the energy efficiency of mobile sensing applications. First, we present update protocols which are able to exploit context predictions for reducing the number of transmitted context update messages and trading off context accuracy vs. energy consumption. Then, we extend our approach and show how knowledge about a user's future behaviour can be used to find the optimal update schedule for both sensing and communicating context data given hard bounds on the energy consumption on a mobile device.

We have implemented and validated our context prediction models in detailed experimental evaluations using synthetic and real-world context data. The results of our experiments demonstrate the effectiveness of our concepts for enhancing the accuracy and expressive power of predictions, as well as for increasing the energy efficiency of context-aware mobile systems.

Zusammenfassung

In den letzten Jahren haben sich kontextbezogene Systeme von einer reinen Forschungsdisziplin hin zu einem allgemein anerkannten Prinzip für die Entwicklung moderner mobiler Systeme entwickelt. Mobile Anwendungen, die sich automatisch an den dynamischen Kontext eines Benutzers anpassen und so die Einfachheit der Interaktion von Mensch und Maschine verbessern können, sind heutzutage weit verbreitet. Mit gegenwärtigen kontextbezogenen Anwendungen wie bspw. Restaurant-Finder oder mobilen Touristenführer ist es jedoch nur möglich Nutzer hinsichtlich Ihrem gegenwärtigen Verhalten zu unterstützen. Als nächste Stufe im Forschungsbereich kontextbezogener Systeme wird dagegen an proaktiven Anwendungen gearbeitet, die nicht nur auf den gegenwärtigen, sondern auch den zukünftigen Kontext von Benutzern intelligent reagieren können: Intelligente Häuser, deren Steuerung an die voraussichtlichen Handlungen Ihrer Bewohner angepasst wird; Anwendungen im Bereich sozialer Netzwerke, die Nutzer über Orte benachrichtigen, welche von Ihren Freunden als nächste besucht werden; Personalisierte mobile Empfehlungssysteme, die über ortsabhängige Ereignisse und Angebote informieren, welche hinsichtlich der Tagesabläufe Ihrer Nutzer relevant sind.

Die Entwicklung von geeigneten Technologien zur Kontextvorhersage um solche Anwendungen Wirklichkeit werden zu lassen stellt jedoch eine große Herausforderung dar. Ein Grund hierfür ist, dass zukünftige Kontextinformationen in den rohen Kontextdaten versteckt sind, die Menschen in der realen Welt zurücklassen und somit nicht einfach zugreifbar sind. Daher werden Verfahren zur Kontextvorhersage gebraucht, die in der Lage sind charakteristische Verhaltensmuster aus erfassten Kontexthistorien zu erkennen. In der gegenwärtigen Forschung fehlen jedoch Verfahren, die genaue und ausdrucksstarke Vorhersagen möglich machen und dieses Wissen ausnutzen können um kontextbezogene Systeme zu verbessern. Infolgedessen ist das ganze Potential kontextbezogener Systeme noch nicht volllständig ausgeschöpft worden. Um dieses Problem zu beheben, steuern wir in dieser Arbeit neue Algorithmen und Modelle bei zur Vorhersage von beliebigem zustandsbasiertem Kontext wie bspw. die Orte und Aktivitäten eines Nutzers. In dieser Hinsicht leistet diese Arbeit die folgenden Beiträge.

Im ersten Teil dieser Arbeit stellen wir ein neues Kontextvorhersagesystem vor, welches auf Grundlage von statistischer Modellierungskonzepte ein für Computer verarbeitbares Modell des Nutzerverhaltens automatisch lernen und Kontextvorhersagen ableiten kann. Mit unserem Kontextvorhersagesystem adressieren wir Schwachstellen hinsichtlich Genauigkeit und Ausdruckstärke der Vorhersage, die wir bei gegenwärtigen Ansätzen ausgemacht haben. Um die Vorhersagegenauigkeit gegenüber bestehenden Verfahren zu verbessern, wird ein neuer Kontextprädiktor entwickelt, der die Beziehung zwischen Kontextänderungen und vom Nutzer ausgeführten Aktivitäten lernt um bedingte Abhängigkeiten für die Vorhersage ausnutzen zu können. Darüber hinaus erforscht diese Arbeit den Einsatz von Algorithmen aus dem Bereich Model Checking um ausdrucksstarke zeitabhängige Vorhersagen berechnen zu können und so die limitierte Ausdrucksstärke existierender Ansätze zu verbessern. Mit Hilfe der Algorithmen und Modelle, die im ersten Teil dieser Arbeit entwickelt werden, erhöhen wir so die Menge und Genauigkeit an Information, die proaktive Anwendungen für die Vorhersage von zukünftigem Kontext zur Verfügung gestellt wird.

Im zweiten Teil dieser Arbeit wird das Augenmerk auf neue Ansätze zur Kontextvorhersage gerichtet um die Performanz mobiler sensorgetriebener Anwendungen zu optimieren. Diese Anwendungen stellen eine neue Klasse mobiler Systeme dar, welche im Fokus gegenwärtiger Forschungsarbeiten stehen um kontinuierlich Kontext auf einem mobilen Gerät zu erfassen und über drahtlose Kommunikationskanäle an interessierte Konsumenten weiterzuleiten. Da mobile Datenkommunikation für mobile Geräte einen beträchtlichen Energiemehraufwand bedeutet, entwickeln wir in dieser Arbeit neue vorhersagebasierte Updateprotokolle um die Energieeffizienz dieser Anwendungen zu verbessern. Als Erstes stellen wir Updateprotokolle vor, die Kontextvorhersagen ausnutzen um die Anzahl der übertragenen Kontextupdates zu reduzieren und den anfallenden Energieverbrauch gegenüber der erzielten Kontextgenauigkeit abwägen zu können. Zusätzlich wird ein erweiterter Ansatz entwickelt, der Kontextvorhersagen ausnutzt um eine optimale Entscheidung zu finden, wann Kontext erfasst und verschickt werden soll, wenn feste Schranken hinsichtlich des Energieverbrauchs auf mobilen Geräten gegeben sind.

Die in dieser Arbeit beschriebenen Kontextvorhersageansätze wurden implementiert und in ausführlichen Experimenten mittels simulierten und echten Datensätzen bewertet. Die Ergebnisse der Experimente belegen die Wirksamkeit der Ansätze um die Genauigkeit und Ausdrucksstärke von Vorhersagen verbessern und die Energieeffizienz kontextbezogener mobiler System erhöhen zu können.

Part I.

Introduction and Background

Chapter 1

Introduction

1.1. Motivation

Challenged by the increasing complexity of today's software systems and physical environments, new technologies are required which seamlessly integrate with human life. The advent of context-aware computing has paved the way for a new generation of personalised and ubiquitous applications [ST94]. These applications exploit awareness of a user's context to provide intelligent services and promote unobtrusive interactions of humans with their surroundings. Thus, users of context-aware applications are promised a unique and convenient experience: no manual input is necessary to communicate their aspired goals and intentions with computer systems. This has led to a vision where computer devices of various forms and shapes, e.g., smartphones, public displays, and wearable devices, can automatically recognize the user's needs and adapt to them.

In order for this vision to come true, context-aware applications have to gain accurate knowledge about the context of users and the environment in which they are acting [SBG99]. Hence, since the early days of context-aware computing, a major interest has been in techniques to extract and infer meaningful information that are relevant to a user's context. In the past, research has primarily focused on approaches for recognizing a user's real-time context, including information about his physical state (e.g. location, activity) [TIL04], emotional situation (e.g. stress level) [LRC+12], social state (e.g. in a meeting) [EPT+06], or parameters of the surrounding environment (e.g. light, sound, etc.) [BBHS03]. Based on these technologies, novel human-centred applications such as restaurant finders or mobile tour guides could be developed that rely on information about users' instantaneous context [SGP+05]. This enables the provision of effective context-aware services to support humans in their current environment, e.g., providing a list of recommended restaurants which are close to the user's current position. As a next stage in context-aware computing which goes even one step further, not only the

user's current context, but also his future context shall become relevant to inform new kinds of applications and services [PNF05].

Future context information is seen as a key to facilitate novel proactive context-aware systems that are able to incorporate the prospective behaviour of users [May05]. These systems are envisioned to exploit predictions of the future situation of users, e.g. where they might be going and what they might be doing, for improving the experience of human-computer interactions and enabling new ways of how technology can foresee the needs of humans at any time scale. For instance, in mobile advertising, knowledge about future context enables enhanced information campaigns, advertising location-dependent offers to people whose forthcoming travel routes are close to points-of-interests such as shops or restaurants [PP09]. This allows customers to be informed with personalized offers linked to their future activities and mobility habits, thus increasing the relevance and usefulness of information seen by them. Further applications of context predictions to enhance the proactive behaviour of computer systems can be found in a wide range of different scenarios. For example, in the field of ambient intelligence, smart homes have emerged as the idea of automated living environments which can seamlessly adapt to the life habits of their residents [RDB07]. In order to increase the comfort of living and reduce the costs of operating the house, knowledge about the residents' current and prospective actions is vital to control the ambient environment in a way which does not interfere with their typical usage patterns [RRD06].

However, while the great potential of proactive computing systems has been often discussed among researchers, many of the envisioned application scenarios cannot be realized yet. In particular, effective context prediction techniques are missing which are able to gain knowledge about users' future context patterns required to enable proactive behaviours. This is due to the fact that, in contrast to present and past context data, future context is a piece of hidden information which is not directly visible from sensor output. Present context information can be provided by currently available commercial or academic context recognition systems, e.g., location positioning systems such as UbiSense [SG05] or activity recognition frameworks such as Opportunity [RLFC11]. These systems process and interpret raw signals from a set of heterogeneous sensors, e.g., GPS, infra-red, accelerometers or microphones, to gain a real-time understanding of the current situation of humans. Past context information is accumulated in temporal databases or knowledge repositories, where changing contextual behaviour is recorded over a period of time [HLA+04]. This allows for advanced context-aware services involving histories of time-stamped context data, e.g., queries for past trajectories of moving objects [LDR08]. However, future context data is encoded in the typical habits and routines of human behaviour and therefore much more difficult to obtain. It is neither available from the immediate output of sensors nor from collections of past context records. Therefore, the field of context prediction and the development of sophisticated prediction techniques is still an open research problem.

A great challenge in predicting context data is due to the intrinsic characteristics of human behaviour. Since our everyday choices are variable and transient, there is a high degree of uncertainty involved in human actions which makes context prediction a difficult and complicated task. Therefore, predictions of future context cannot be simply derived from deterministic rules, but more complex methods are required which are based on an inherent measure of uncertainty to learn and represent a model of the user's behavioural patterns. The choice of a suitable prediction model and its underlying properties has therefore large implications on the predictability of human behaviour. In prior research, the focus has been on fundamental aspects of context prediction systems such as system architecture design, context inference, and basic prediction algorithms [May04]. However, even though the idea of context prediction for designing proactive computer system has gained importance since then [PNF05], there has been only little progress in designing more effective context prediction approaches and their exploitations in specific contexts. In this thesis, we identify and overcome several major limitations of current prediction systems in order to help proactive applications become more sophisticated and gain more practical relevance.

First, in order to increase the satisfaction of users with proactive applications, context predictions systems must provide predictions with a high degree of accuracy for their actual occurrence. However, while context variables of different types (e.g. a user's activity and location) may reveal useful correlations, existing context predictions systems often do not model such latent relationships. Incorporating dependencies among different context types can be a relevant discriminating factor for resolving uncertainties and achieving more accurate predictions. Second, the type of forecasts which are feasible with current prediction system are highly restricted in query semantics. Currently, only predictions of the next most likely context state are supported (e.g. the user's next location). In order to allow for predictions which go beyond the next context state, new methods are required which are able to increase the expressiveness and range of possible forecasts that can be derived from context histories. Third, new application domains arise where context prediction can play an important role to improve the behaviour of these systems. In particular, mobile sensing applications (e.g. mobile social networking [MPF$^+$10]) are becoming increasingly popular as a research area, but face technical obstacles for a wide-spread deployment in reality. Since these applications are designed to transfer continuous steams of context data from the user's mobile device to remote subscribers, high energy costs are incurred that are a critical factor for the usability of mobile devices and thus the end users' experience. The great potential of using context prediction for optimising the energy costs of these applications has not been addressed in current state-of-the-art.

As this discussion shows, significant challenging research problems can be identified which have not been sufficiently addressed in prior research. In this thesis, we investigate these problems in detail and propose new context prediction methods to solve them. A detailed discussion of the contributions of this thesis is subject to the next section.

1.2. Contributions

In this thesis, we contribute a comprehensive approach for the prediction of mobile user context to improve the proactive behaviour of context-aware applications. To this end, we propose a context prediction system that is based on novel statistical methods to forecast state-based user context. State-based context data represents a powerful context model in the focus of current research to describe a wide spectrum of meaningful user behaviours. Developing an effective context prediction system that supports state-based user context therefore enables context forecasts in various application domains, addressing user activities (e.g. sitting, biking, walking), symbolic location information (e.g. home or office), social or emotional status (e.g. in a meeting), and even combinations of those (e.g. sitting in front of the TV at home).

However, in order to develop such an approach, a number of significant challenges need to be addressed. These challenges arise from the need for a set of key characteristics that proactive applications require to obtain knowledge of future context information. A fundamental problem is to develop a context prediction system which cares for accurate predictions, enables applications to pose queries for future context with expressive semantics, and provides the ground for prediction-based mechanisms to overcome current limitations of context-aware applications. In the following, we elaborate on each of these dimensions, explaining the underlying problem in detail so as to state our concise contributions.

1. **Prediction Accuracy**: The quality of service delivered by proactive applications is strongly depending on the prediction accuracy of a context prediction system. Since these applications are supposed to take proactive actions on behalf of the user, reliable context predictions are essential for enabling a positive user experience. While current prediction systems incorporate limited information about a user's past behaviour into a prediction (e.g. the sequence of most recently visited locations), the instantaneous decisions taken by users often depend on changing external conditions which are not reflected by these patterns. Activity models which are designed to unveil the conditions and constraints of a user's actions appear as an ideal choice to close this gap. However, the integration of information from activity models into context predictions systems has not been explored yet to help deliver more reliable forecasts. Therefore, the first problem addressed in this thesis can be phrased as: how can we integrate and exploit models of a user's activities to increase the accuracy of context predictions over current prediction approaches?

   To address this challenge, we propose a new approach to inject knowledge from flow-based models of user behaviour into context prediction systems. These models reveal the flows of activities a user performs in process-oriented application domains, e.g., typical work routines in hospitals. We propose a new predictor, the so-called flow-based predictor, which combines information about activity

executions with the user's context changes in a statistical model of the user's behaviour. This model is able capture latent conditional dependencies for prediction where the user's next context change is related to his attached flow of activities (e.g. when moving to a room to perform the next work item). In order to realize this approach, we develop a new bi-variate Markov predictor which encodes both information sources in a probabilistic state transition system. For this purpose, we propose a learning algorithm, which is explicitly designed to reflect the correlation of activity and context information, and present prediction algorithms to efficiently traverse the probabilistic search space for determining the most likely upcoming context states. Our evaluations show that the flow-based prediction scheme significantly increases the prediction accuracy when exploiting activity information as a discriminating factor.

2. **Prediction Expressiveness**: The value of a context prediction system is strongly influenced by the range of queries it is able to answer. In order to support the rich information needs of proactive applications, a powerful prediction interface is required. In particular, an expressive prediction query language that allows application to pose flexible queries with well-defined semantics is vital. Unfortunately, current context prediction system only cater for simple queries, as just next-state predictions are supported (e.g. predictions of a user's next location). Further methods for answering time-dependent queries and exploring context occurrences beyond the immediate state successors at an arbitrary future time are not foreseen. This severely limits applications in many scenarios where more advanced context predictions are required. Therefore, the second main problem studied in this thesis is: how can we devise a context prediction system that supports more expressive queries than possible with current state-of-the-art approaches?

   As a response to this challenge, we develop PreCon, a novel context prediction system which allows for expressive predictions of a user's context changes. PreCon is based on a temporal-logic query language, which entails various temporal operators to query for future context with well-defined formal semantics. Due to the expressive power of temporal logics, predictions with time-dependent constraints become possible, e.g. whether a user x will eventually reach his home within a time horizon of t minutes. In order to accomplish this, PreCon improves stochastic model checking techniques from the field of formal system analysis. Specifically, the user's behaviour is represented as a Semi-Markov Model which encodes a probabilistic state transition system augmented with state dwell time information. Context prediction is then performed as a stochastic inference process over the Semi-Markov Model to infer the probability of satisfying temporal-logic formulas with time constraints. In our evaluation, we apply our approach to real-world context data and demonstrate the effectiveness of our approach based on metrics from information retrieval. PreCon is the first context prediction approach which explores the combination of statistical methods and logic-based formalisms to

express and compute forecasts of human behaviour.

3. **Prediction-based Cost Optimization in Mobile Systems**: In recent years, mobile sensing applications have emerged (e.g. mobile social networks), which require to constantly sense and send streams of context updates over ubiquitous wireless networks to interested subscribers. However, due to the limited capacities of the devices' batteries, energy expensive operations are critical for the batteries' lifetime and may significantly impair the usability of mobile devices. A fundamental challenge in these systems is therefore to achieve a good trade-off, i.e., one that does not overspend the device's energy budget, but nevertheless guarantees a high context accuracy to interested consumers. In order to improve this trade-off, predicting a user's future behaviour provides means to acquire knowledge about a user's context with much less energy overhead. Specifically, context prediction can be an interesting technique to estimate a user's yet unknown context. However, the opportunity of this approach for increasing the energy efficiency of mobile sensing systems has not been explored in prior research. Therefore, as a third main problem studied in this thesis we investigate: how we can optimize the costs of acquiring streams of context data on mobile devices using novel context prediction approaches?

In this thesis, we tackle this challenge by the design of protocols and algorithms, which make use of tailored context predictions to optimize the energy/accuracy trade-off in mobile sensing applications.

First, we develop novel update protocols for the efficient distribution of streams of context data on mobile devices based on integrated prediction mechanisms. Specifically, we propose time- and deviation-based update protocols, which limit the inaccuracy of the context data experienced by remote consumers, and combine them with prediction-based update strategies to provide an update-to-date view on a user's context in spite of missing update messages. For this purpose, we devise and compare four Markov predictors that can be used to accurately predict the current context state of a user based on the last context update which has been received. Since accurate predictions reduce the need for continuous transmissions of context changes, our prediction-based update protocols are able to significantly lower the energy consumption on the sending as well as the receiving device.

Second, as an extension to these protocols, we propose a more rigorous approach, which not only reduces the energy consumption, but gives hard guarantees about the energy required on mobile devices for sensing and communication of context data. We approach this problem as a constrained optimization problem, where the accuracy of the context data that is delivered to remote consumers shall be maximized while the amount of consumable energy is strictly limited by a given energy budget. We formalize the problem as a Constrained Markov Decision Process (CMDP), and propose

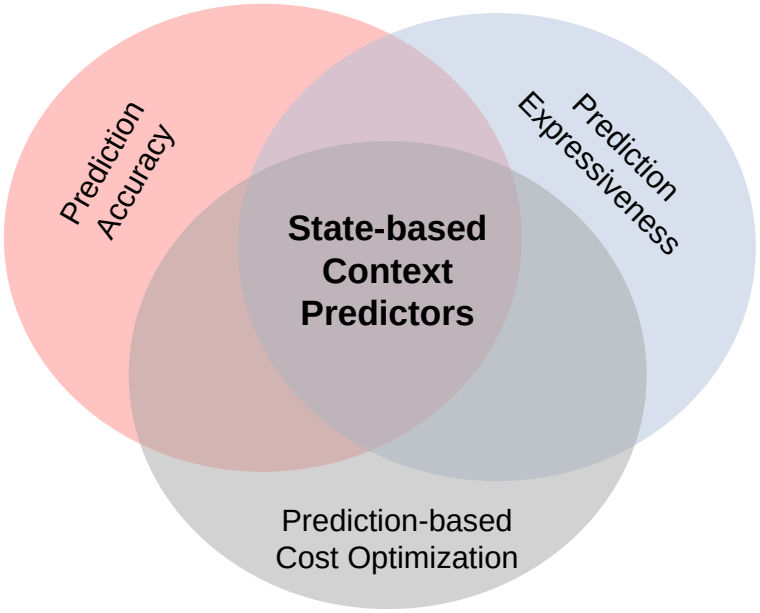


Figure 1.1.: Venn diagram of thesis contributions

a statistical model of the update process for which we incorporate information about a user's predicted context changes. Based on this model, we find an optimal configuration of our update protocols which guarantees the most effective update decisions within the given energy limit. In our evaluation, we show for real-world traces of user behaviour that even for small energy budgets we can achieve a high context accuracy. This means that significant energy savings can be achieved on the device, while the loss of quality for context-aware applications in terms of the negative effect on the context accuracy is minimal.

The main contributions of this thesis are visually summarized in the Venn diagram shown in Figure 1.1. The core of our contributions are novel state-based context predictors, which are devised to address the three problems stated above: improving prediction accuracy, enhancing prediction expressiveness and using predictions to optimize the costs of mobile systems. We hence contribute novel effective solutions to fundamental problems in the field of context prediction, ranging from universal prediction models to tailored prediction-based approaches for improving the performance of mobile systems. As a result of this research, the context predictors developed over the course of this dissertation can be exploited to increase the proactiveness and efficiency of mobile systems in a broad range of different scenarios. As part of this thesis, the contributions have also been described in different publications [FHH10, EFH+09, FHR11, FHR12, FBHR12].

1.3. The ALLOW Project

The research of this thesis has been conducted within the research project ALLOW - Adaptable Pervasive Flows [HRKD08]. ALLOW is a Future and Emerging Technologies (FET) project funded by the European Commission under the 7th Framework Programme. The project was running from February 1st 2008 to July 31st 2011 and involved six universities across Europe.

The goal of the project has been the development of a new programming paradigm for human-oriented pervasive applications. Key to this approach are behavioural modeling concepts (called flow models) that allow for formalizing and describing the activities of humans in the real world. Based on these models, pervasive technical systems are enabled to adapt automatically and seamlessly to humans involved and embedded in them, explicitly supporting people in achieving well-defined goals in dynamically changing environments and contexts.

To achieve the goals of the project, activity recognition and context prediction have been considered as key approaches to provide pervasive application with rich knowledge about the dynamic context of humans. While activity recognition is concerned with the extraction of higher-level activities from low-level data signals (e.g. sound or acceleration) in real-time, context prediction is a complementary concept and studies the evolution of this behaviour over time. As a consequence, context information of different temporal scales - the past, present and future - can be derived when context recognition and prediction are applied simultaneously, thus yielding a comprehensive picture of situations that are or become relevant to the user.

As a result of this thesis, we have contributed new machine-processable behavioural user models and prediction algorithms to the ALLOW vision. These concepts have been integrated into the overall ALLOW system architecture to support proactive designs of applications in domains such as logistics or health-care which have been in the main focus of the project. Aside from these scenarios, the ALLOW concepts can be applied to any domain where pervasive applications are driven by (future) changes in human behaviour.

1.4. Structure of Thesis

This thesis is structured as follows.

In Chapter 2 important background information is given about previous research into context prediction. As part of this chapter, the role of context prediction for designing proactive computer systems is discussed. This includes a detailed description of application domains in which context prediction can improve the behaviour of computer systems. Further, a range of available statistical data analysis methods is

reviewed, which we can be exploited as a foundation to deal with different kinds of prediction problems.

Given the required background information to understand the problem domain, we then describe the contributions of this thesis along two major parts which both involve several chapters.

The first part of this thesis is focused on the design and implementation of novel context prediction models with high expressiveness and accuracy. Initially, we present in Chapter 3 the architecture of our context prediction system and discuss different components which are involved in predicting context data. Based on this architecture, we introduce in Chapter 4 a new context predictor for enhancing the accuracy of context predictions by exploiting the relationship of flows of human activities and associated context changes. Then, we present in Chapter 5 a new context prediction system that is inspired by model checking approaches to enable new forms of expressive context predictions.

The second part of this thesis studies how context prediction can be exploited to improve the energy efficiency of mobile sensing applications, a class of mobile systems in the current focus of research. Initially, we present in Chapter 6 a system model of mobile sensing applications and analyse the energy costs on mobile devices for communicating frequent updates of context data over wireless communication networks. Then, we propose in Chapter 7 a prediction-based update protocol which reduces the costs for sending context updates by anticipating context changes at a consumer. Furthermore, we extend this approach in Section 8 and develop predictive update protocols to satisfy hard energy limits on a mobile device for sensing and communicating streams of discrete context data.

Finally, we summarize the contributions of this thesis in Chapter 9 where the key results are listed. Moreover, we discuss in Chapter 10 suggestions for future work in the field of context prediction, which extend on the contributions of this thesis.

Chapter 2

Background

In this chapter, we discuss various important concepts from the area of context-aware computing that form the background of this thesis. First, we highlight in Section 2.1 significant advancements in context-aware computing which have paved the way for research into context prediction systems. In Section 2.2, we then explore a range of different application areas, in which knowledge about future context can be exploited to enhance the behaviour of computing systems and experience of users. Thereupon, we describe in Section 2.3 the relation of context recognition and context prediction using a layered architecture of a proactive context-aware system. Subsequently, we review in Section 2.4 prevailing data prediction methods that represent well-known tools for statistical data analysis. Finally, we summarize in Section 2.5 the main facts from our background study and discuss their relevance for this thesis.

2.1. Trends in Context-aware Computing

The vision of context-aware computing has led to a new view on the design of computer systems and applications. As proposed by Schilit et al. [ST94], context-awareness is defined as the ability of a mobile user's application to perceive and respond to various stimuli from the environment. While former software systems showed no relation to the surrounding environment in which computers are embedded, context-aware computing thus promotes a new way of improving the interactions of computers with the physical world. Therefore, Schilit et al. stated that context-aware applications can be characterized as software that adapts according to the location of mobile users, the collection of nearby people, hosts, and accessible devices [BNSW94]. For instance, the Active Badge system [WHFG92], an infra-red based indoor location system, is considered to be the first context-aware system on top of which applications such as contextual reminders [BNSW94] have been developed.

Central to these applications is the notion of context, whose meaning gradually expanded over the years to cover a wide range of different aspects. Initially, Schilit et al. proposed a classification scheme and distinguished between different categories of context; these are the user environment, physical context and computing context [BNSW94]. Further, Schmidt et al. have stressed the importance of context information beyond location and advocated a wider understanding of context including information from a range of different sensors, e.g., audio, motion or bio-sensors [SBG99]. Dey proposed a more general definition of context, which is nowadays widely accepted among researchers. He referred to context as any information that can be used to characterize the situation of an entity, which is relevant for the interaction between a user and applications [Dey00]. This discussion around the notion of context shows that from the very beginning of research in this area there has been a great interest in extending the scope and meaning of context information.

In recent years, a new trend towards the discovery of more sophisticated, complex context information could be observed. Specifically, a lot of advancement has been made in the field of activity recognition, which aims at the detection of higher-level user activities [AR11]. Early activity recognition systems required users to wear wireless sensors distributed across several parts of the human body, e.g., wrists and hips [BI04]. Due to their complex and expensive setup, these systems have been successfully applied to specific settings only, e.g. maintenance assistance systems [KWK+09], and found to be of limited relevance in more natural everyday situations. Therefore, the latest research has concentrated on commercial off-the-shelf mobile phones as underlying sensor hardware, and it has been shown that human activities such as walking and running can be accurately recognized by smartphones with on-board sensors such as accelerometers or microphones [KWM10]. As a consequence from the steady progress in the field of activity recognition, the knowledge available to ubiquitous computing systems has expanded and facilitated new context-aware applications with rich information about the user's behaviour. For instance, in application domains such as health-care, novel applications have been developed to monitor and assess the activities of patients to give medical advice or automatically notify caregivers about recognized anomalies in a patient's behaviour [TF08, SARK09].

Beyond these techniques, further directions are currently pursued to discover useful context information and better understand human behaviour. While context-aware systems have often relied on technologies for acquiring the users' current context, the temporal evolution of this context has also been considered to be a valuable source of information [May05]. This allows for making assumptions about the user's prospective behaviour, thus enabling computers to support users not only in their current environment, but also when moving through distinct contexts (e.g. related to their locations or activities) over time. Consequently, as a latest trend in context-aware computing, context prediction methods are studied for improving the proactive behaviour of context-aware applications. For this purpose, researchers analyse traces

of context data to discover characteristic patterns in the users' behaviour. To support the goals of this research, repositories such as CRAWDAD [KH05] have emerged which provide access to human context traces that have been recorded as part of various user studies. The availability of such real-world data has intensified the interest in context prediction among researchers with different backgrounds. This research area is driven by the idea of novel proactive applications that can take advantage from such predictions. A detailed discussion of such proactive applications and the benefits that they can provide is subject to the next section.

2.2. Applications of Context Prediction

Context prediction is seen as a basic enabler of proactive behaviour in various application domains. Proactivity describes a special feature of computing systems, and characterizes the ability of applications to take the initiative on behalf of users [Sig08]. In conjunction with context-aware systems, this can be understood as the ability of performing adaptations in expectation of a course of future actions and situations of the user. Compared to a reactive system, the user experience can greatly benefit from such an application design, since the prospective steps and future needs are directly reflected in the decisions of those applications.

In the following, we discuss various applications areas where knowledge about the user's future behaviour is of major benefit. Even though the list is not non-exhaustive with respect to the many potential use cases that can be envisioned with current or even future technologies, it will clearly demonstrate in which ways a proactive design of context-aware systems can improve the underlying system performance. For this purpose, various performance metrics are discussed which relate to both the technical aspects of systems (e.g. reduced energy consumption of operating smart homes) as well as the end user experience (e.g. retrieval of more relevant personalized information), where the availability of future context data enables improved behaviours of context-aware computing platforms.

Wireless Networks. Wireless networks provide mobile users seamless access to information anytime and anywhere. From a technical point of view, wireless data connections are established among the users' devices and the nearest wireless access points (in case of WiFi) or base stations (in case of e.g. GPRS or UMTS) *. In order to maintain these connections in the face of the user's mobility, during which the user may leave the communication range of an access point, efficient handovers are required. As part of such a handover, the state associated with the active wireless connection has to be transferred between two neighboured access points, which is a critical step for the data connection quality. It has been shown that such handovers can incur significant delays

*For the ease of explanation, we refer in the following to wireless access points only, while the same principle equally applies to base station handovers

when the transfer is initiated only after the physical association has taken place [MSA03]. This negatively affects the quality of the communication service, especially in case of applications such as voice or multimedia applications where the users often keep permanent connections while moving through the network. To alleviate this problem, it has been shown that a proactive approach can significantly reduce these delays and support more effective handover management [MSA04]. The idea behind proactive handovers is to transfer the session information associated with the connection already ahead of the association time to other access points. In order to guarantee a scalable handover management and minimize the number redundant transfers, predictions of the user's mobility behaviour are used to identify the access stations with which he most probably connects next. This has shown to achieve significant reductions of the handover latency in the scale of an order of magnitude [MSA04]. Consequently, the availability of context predictions in these scenarios enhances the management of complex decentralized wireless systems to guarantee users a seamless experience of always-on connections.

Mobile Advertising. With the steady increase of mobile phone usage, mobile advertising has become a new marketing tool to reach a large number of potential customers which might be interested in novel products or services [Kru11]. However, advertisements are often experienced as spam by receivers due to the large amount of information that is not relevant to them. Therefore, new ways of how to provide potential customers with personalized advertisements that better suit their individual interests are researched. Specifically, it has been proposed to enhance the effectiveness of advertisement based on information campaigns, where the content of advertisements should match the context of mobile users closely [PP09]. For instance, the users' visited locations reveal important information to enable advertisements which are bound to certain spots, e.g., for sending coupons to people living in close neighbourhood of the shops providing these offers. However, most often these advertisements are useful only prior to a potential purchase decision, so that interested buyers can become aware of these offers and take purchases into consideration. Therefore, not only the current, but also future context can provide valuable hints for placing more effective advertisements. For instance, knowing the typical routes of people, special offers can be recommended that are offered by shops which are in close travel time to those routes. The knowledge about these routes can be gained from predictions about the sequence of locations visited by users at different days of a week. This form of mobile advertising has great potential in increasing the revenue of businesses and user satisfaction, thus establishing a win-win situation for both, advertisers and consumers [PP09]. The prospective of how successfully such an advertisement strategy can become is therefore strongly dependent on sophisticated context prediction capabilities.

Smart Homes. The vision of transforming peoples' homes into intelligent environments has led to the development of smart homes [RRD06]. Smart homes are enriched with sensor technologies to allow computers to effectively control various aspects that affect

the in-house living conditions, e.g. heatings, lightings, etc. Since the operation of houses is a complex of task for humans, smart home technology can thus provide relevant support to increase the living comfort and improve the resource consumption efficiency. Among the benefits, improved safety, reduction of costs, or ease of use are considered to be the ones most valued by the house inhabitants [RBB+03]. However, in order to automatically control and adapt the house environment, the specific lifestyles of the house's inhabitants need to be incorporated [RDB07]. This is to ensure that the house environment is proactively brought to a state which is aligned with the everyday needs and usage patterns of its residents. For instance, it has been shown that for controlling actuators such as lights, fans, or air-conditioner in an automatic manner, a smart house needs to recognize the absence of its inhabitants and anticipate when they are most likely to come back [RRD06]. Specifically, to regulate the temperature in various rooms with respect to the habits of the house's inhabitants, the air conditioning system can be deactivated during times of their absence, but must be activated already before these rooms are expected to be used. Therefore, location and movement predictions can reveal important information to identify at what times rooms are most likely accessed in the house [RRD06]. This way, the manual effort can be significantly reduced to relieve users from technical tasks of configuring their house environment. Consequently, context predictions are relevant to seamlessly adapt technical systems which intervene with the daily lives of people to automate human tasks.

Information Hoarding. The proliferation of mobile devices has changed the way in which information can be accessed, enabling us to be continuously connect with the Internet from almost everywhere regardless of our current location. However, since mobile devices are limited in available resources such as their energy capacity or available network bandwidth, novel strategies are required for mobile information access that impose less burden on the resources of mobile devices. In particular, the concept of information hoarding [KR01b, BMR04] has been proposed, where the idea is that required information could be proactively downloaded during times of high-bandwidth wireless data connections (e.g. in case of WiFi) and stored on the device for later usage. Since with this approach most of the network communication activities are concentrated on times of cheap data connections, the energy consumption on mobile devices can be reduced and information can be provided with short response times. Key to this approach is the accurate prediction of information items that can become relevant to the user, given constraints about the amount of data which can be hoarded on a mobile device. As in mobile computing scenarios information access has found to be location-dependent, a hoarding approach can take advantage of a user's predicted future locations to increase the cache hit rate and thus make the process of information hoarding more effective [KR02]. Based on the user's predicted destinations, the information objects with the highest access popularities at these locations can then be transferred to the device. In order to improve the predictions, it has been show that further domain knowledge, e.g. route trajectories as being managed by navigation systems, are useful to identify the data items with are accessed during

future user trips [KR01a]. Consequently, to decrease the cost of information access over wireless networks based on an effective information hoarding strategy, accurate context predictions are of great value to optimize the pre-fetching decisions.

Opportunistic Communication. Traditionally, mobile communication is realized as an infrastructure-based service provided by mobile network operators. However, with the standardization of wireless ad-hoc interfaces (e.g. Bluetooth or WiFi) and their integration into commodity mobile phones, alternative ways of information delivery have become possible. Since cellular networks are becoming increasingly overloaded due to the large volumes of mobile Internet traffic consumed by users, new strategies for reducing the load on cellular traffic are at the centre of current research [HHK+10]. Opportunistic communication is a new communication paradigm which is based on the idea that data can be directly exchanged among mobile devices within local proximity using short-range radio communication. This significantly reduces the dependency on a conventional communication infrastructure, since the process of information dissemination is performed completely autonomously or at least partly managed by the mobile devices themselves [HCY08, BDR12a]. However, as human behaviour is characterized by a high degree of mobility, human-carried mobile devices permanently enter and leave the mutual communication range of their wireless radio adapters in these scenarios. As a consequence, sophisticated routing strategies have to be developed which can guarantee successful messages delivery based on opportunistic contacts of nodes in the network. In order to achieve scalability and avoid flooding of data packets, selective routing decisions have to be made to judge if a potential carrier is a good candidate to help deliver a message to its final destination. For this purpose, it has been proposed to predict encounters of mobile nodes at specific locations, which allows for sending messages to selected nodes with a high meeting probability [LFC06]. Beyond co-location patterns, also predictions about the social relationship [DH07] and the underlying social network [HCY08] of message receivers have shown to speed-up the information propagation process through the network. This is strongly different to classical routing in infrastructure-based networks, since opportunistic communication cannot rely on static network links and pre-configured routes. Therefore, predictions are used to compensate for this lack of knowledge and it has been demonstrated that such an approach can greatly relieve the load from the cellular network infrastructure [HHK+10, BDR12a].

2.3. Architecture of a Proactive Context-Aware System

In the section, we discuss a layered view on the processing of context data in context-aware computing scenarios. For this purpose, we introduce the architecture of a proactive context-aware computing system which requires both recognition and prediction of mobile user context. The architecture is found in state-of-the-art context-aware computing systems [May04], [Sig08] and can be seen as a de facto standard for reasoning about

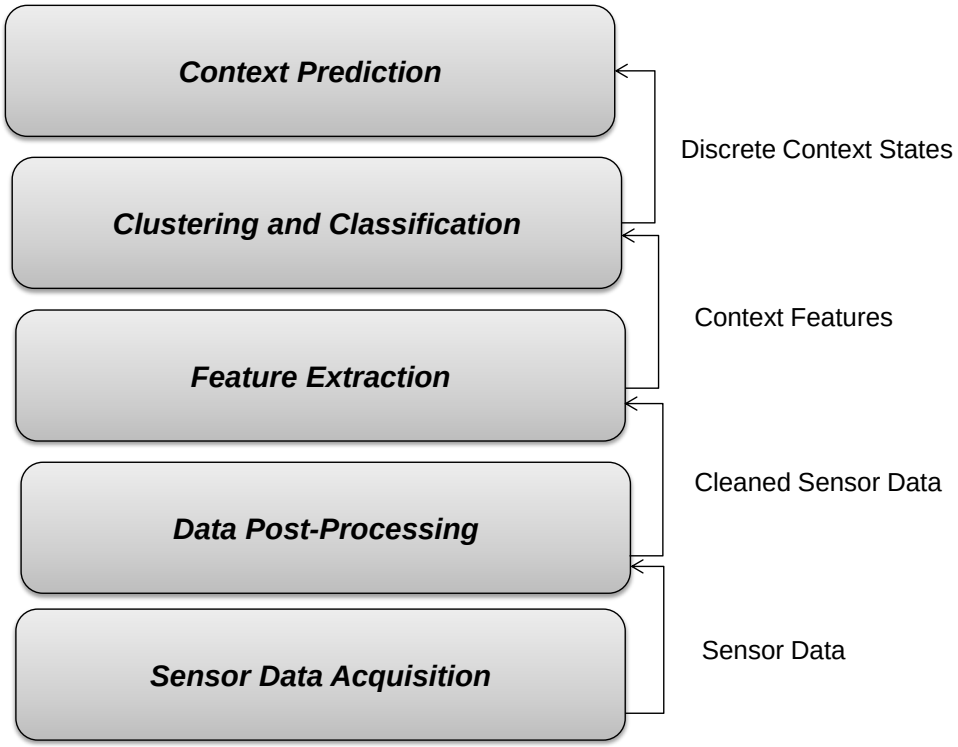


Figure 2.1.: Layered architecture of a proactive context-aware system

context data. In the context of this thesis, a discussion of this architecture is predominantly relevant for two reasons. First, it shows that the data obtained from sensors as raw values needs to undergo various transformation steps before it becomes useful for the task of context prediction. This will clarify on which level of abstraction our context prediction system is operating. Second, the architecture precisely describes the relation between context recognition and context prediction. Since both are concerned with tools for the processing of context data, this will stress the need for additional analysis methods beyond the functions implemented in the context recognition component to gain predictive capabilities.

As shown in Figure 2.1, the architecture is composed of different layers of context data processing services. From an abstract point of view, each service receives context data as input, processes it and then forwards a more abstract representation of the context to the next upper layer. While the raw sensor data is obtained at the most bottom layer, context prediction is at the most upper layer and represents the end of the processing chain. As a result of applying the chain of processing services, the level of abstraction of the context data is gradually increased, so that more knowledge becomes available which is hidden in the raw sensor data. In the following, we describe the functionality

of each layer more in detail following a bottom-up approach.

Sensor Data Acquisition. This layer is concerned with the task of measuring raw sensor data. For this purpose, a wide range of sensors have been developed in recent years which are integrated into commodity hardware modules. For example, modern smartphones such as the iPhone are equipped with various on-board sensors to detect light, proximity, audio, location, video, etc. These sensors can be periodically sampled to obtain a time series of physical measurements. For instance, using a GPS sensor, the trajectory of geographical positions along which the user has moved can be recognized. While the raw data is useful for specific purposes, e.g. for the sake of traffic monitoring, it is often not expressive enough to describe the activities and mobility aspects which are relevant in a human context. In order to gain more expressive meaning, the quantitative measurements in form of numeric sensor values need to be transformed into higher level semantic knowledge of discrete nature (e.g. describing user activities such as walking or sitting). The extraction of this knowledge is subject to the higher layers of the architecture, which look at the patterns contained in the sensor data.

Data Post-Processing. Before the raw sensor data can be analysed for interesting patterns, validation and correction measures need to be applied that make sure that the data is not overly affected by sensing errors. This is necessary because sensing real-world phenomena with mobile devices does often yield inaccurate measurements (e.g. due to faulty hardware, signal disturbances or bad exposition of the sensors to the source of the signal). For instance, the positioning error of GPS is often not better than 5 meters [AG02], while in dense urban settings where not all required satellite signals are visible even much larger errors may be observed. Also, retrieving data samples from the sensors may occasionally fail, so that certain sensor readings may be missing in a series of measurements. Therefore, as part of the post-processing process, the data is cleaned from outliers which are identified as readings with an unnatural behaviour that may imply noisy readings. To counter the outlier effect, interpolation and smoothing techniques are applied where the trend of the data is exploited to adjust existing values and insert averages for missing values in between two neighboured sensor readings. This results in an enhanced time series of sensor values which carries less sensing errors to increase the accuracy of the estimated user context.

Feature Extraction. Given a cleaned time series of sensor values as input, the next step is to study features which are hidden in the data. The motivation behind this is to extract data qualities that make the recognition of higher level context much easier. Typically, features are chosen which describe statistical properties of the data, since this allows to deal with variations in sensor readings which can be observed for real-world user activities. For instance, simple features may be parameters such as the average and standard deviation which shed a light on the distribution of the data, assuming that different context states have unique signatures of such parameters. More complex features may involve computationally intensive task such as the Fourier

transformation, which translates the time series from the time domain into a frequency domain representation. For instance, the frequency domain entropy is a meaningful feature to capture the spread of frequencies in discriminating user activities such as assembly tasks [WLTS06]. Usually, not only a single, but many different features are extracted from the sensor readings to increase the information richness of the data and give insights into different kinds of statistical properties. If the dimension of the feature space becomes too large which may aggravate the data analysis, dimension reduction techniques are applied to keep only the most informative and discriminative features. As result, a vector of feature values is derived for each data sample that is given as input to the next layer for further analysis.

Context recognition. For recognizing meaningful classes of discrete context data such as user activities (e.g. walking or sitting) or semantic locations (e.g. home or office), numeric sensor values need to be transformed to discrete context classes. In order to perform this task, data samples need to be assigned to higher level context states, which embrace features characteristics with the same statistical properties. For this purpose, different learning approaches can be used, among which clustering and classification are the most popular methods for context recognition.

Clustering is a form of unsupervised learning, which means that any prior knowledge about the existence of different clusters (i.e. context states) is not required. For this purpose, clustering algorithms such as the k-means algorithm [Mac67] split the entire data set into subsets of coherent data using suitable distance metrics. The distance metrics are applied to the data feature values, so that members of the same cluster are similar to each other in terms of their feature characteristics. The clusters gained from the clustering process can then be associated with different semantics in human behaviour. For instance, user activities such as walking or sitting can then be identified as clusters with different acceleration signatures.

In contrast to clustering, classification is based on supervised learning which requires more information to be available to perform the recognition of user context [RN09]. In this case, not only the features values associated with each data sample, but also discrete class labels describing the semantics of the data (i.e. name of the context state) has to be provided with the training set. Then, classification methods can be used to learn the relationship of the data features with distinct classes, so that samples of sensor data can be uniquely assigned to one of these classes. While clustering and classification algorithms thus take a distinct approach for context recognition, both provide as output discrete context states which are then available to the context prediction layer.

Context Prediction. The context prediction layer further increases the level of abstraction of the sensed context data. For this purpose, it explores the temporal dimension of the data and discovers patterns that describe sequences of typical context changes. Hence, context prediction can be seen as the study of temporal user behaviour. For the sake of context prediction, the user's behaviour is regarded as a stochastic process which evolves over time. Drawing on the statistical properties of stochastic

processes, the current user behaviour can be extrapolated to identify probable context changes in the future. To accomplish this, the prediction layer requires sophisticated probabilistic reasoning techniques. Therefore, it is designed as a system on its own, consisting of several sub-components with dedicated functions. We will discuss these components in detail when introducing our context prediction system in Chapter 3.

As the discussion of this architecture shows, the interface between context recognition and context prediction is clearly defined in this thesis. Context recognition provides us a time series of the user's context states, which is received by the context prediction component to build a prediction model of temporal context changes. We like to point out that this approach implies that we perform context prediction on the level of discrete user states and not on numeric context data as proposed in [Sig08]. Due to this approach, we are able to forecast meaningful transitions among discrete context states involving significant changes in the underlying sensor data (e.g. transition among the user's home and work place). This is not possible in [Sig08] where only low-level context such as raw sensor values can be predicted that occur within a short time scale.

Also, it is worth noting that the direction of interaction between context recognition and context prediction is not limited to be one-way. The knowledge about the occurrence of future context states can be used to optimize the behaviour of the context recognition layer in turn and reduce the costs of the context recognition process. In this thesis, we will present an approach in Chapter 7 and 8 where context predictions are fed back to the lower layers to enhance the performance of mobile sensing applications, where the process of context recognition faces energy limitations of mobile devices.

2.4. Context Prediction Methods

The application of data analysis and prediction methods as powerful problem solving tools has a long tradition in different fields of computer science. While the invention of these methods often dates back a long time ago, the constant emergence of new data-driven application domains such as context-aware computing has retained their attractiveness and relevance to date. For the purpose of context prediction, there is a wide range of different statistical prediction methods which are potentially useful and interesting.

In this section, we review and discuss existing methods that are commonly used to address prediction problems. In the context of this discussion, we show that each of these methods has been designed with a specific problem in mind, which makes their application most useful in specific application contexts only. In the following, we therefore reflect on their ability to support the prediction of human context, so as to identify the most appropriate statistical prediction framework for the purpose of this thesis. A summary of the discussed prediction methods and their most important features is given in Table 2.1

Prediction Method	Numeric Data	Discrete Data	Temporal Patterns	Data Relations	Hidden Variables
ARMA	X		X		
Regression Analysis	X	X		X	
Classification		X		X	
Bayesian Network		X	X	X	X
Markov Models		X	X		

Table 2.1.: Overview of prevailing prediction methods and their characteristic features

2.4.1. ARMA

Autoregressive Moving Average (ARMA) models are a prominent method for the analysis of time series [BJR08]. They are composed of two different, but complementary parts: the autoregressive (AR) and the moving average (MA) model. An AR model is based on the assumption that the future evolution of a time series is linearly dependent on previous observations. The number of previous terms on which the time series depends is defined by a parameter p, which is also known as the order of the model. For instance, in case of a AR(1) model, the future output is regressed on the last time series sample. In contrast, a MA model does not deal with data that can be directly observed, but handles the effects of noise that is assumed to distort the data. According to this model, the current value of the time series is influenced by q previous white noise terms, where q is the order of the MA model. Each of the noise terms is typically assumed to adhere to a normal distribution with zero means and a known variance. For instance, a MA(1) thus consists of one normally distributed white noise term.

While both models, AR and MR, can be applied independently, ARMA(p,q) describes a combined and hence more expressive model which integrates them into a single prediction function [BJR08]. In order to apply the model to a given application scenario, appropriate parameters for p and q need to be found, which is usually accomplished by inspecting the (partial) autocorrelation function of the time series [Wei05]. Subsequently, the model has to be fitted to given sample data to determine the strengths of the linear dependency. The fitting process is usually based on least-square regression [WMH60]. Then, the fitted model can be used for predicting the evolution of the time series using the most recent observations at current time t.

ARMA is used for continuous time series prediction, where the domain of the random variable is numeric. This is beneficial when the prediction is applied to raw sensor data, such as in case of temperature data which can be fitted to a weather forecast model [BKVR10]. Also, when dealing with financial data such as stocks, ARMA is an effective method to model the time series evolution [Tsa02]. However, in the context of this thesis, where the focus is on context data with semantic labels, ARMA models fall

short to naturally deal with the nature of states. Therefore, models for discrete context data which can be used to reason over state transition trajectories are more meaningful in the scope of our work.

2.4.2. Regression Analysis

Another popular method for statistical reasoning is regression analysis, which allows for studying the relationship of different random variables in data sets [DS98]. For the purpose of regression analysis, different kinds of variables are distinguished. The so-called dependent variable Y models the outcome or effect in the data. The explanatory variables $X_1, X_2, \ldots, X_p$ represent various input features which can be observed. Based on a statistical model of this relationship, the current assignment of the explanatory variables can be used to predict the corresponding value of the dependent variable. As an example, consider the case of a demographic study, where regression analysis can be used to explore the relationship of a population's life expectancy (explanatory variable) with people's year of birth (dependent variable) for predictions about how old people might become in the future [Sha05].

Regression analysis is often based on the variants of linear and logistic regression [DS98], where the former may be applied to continuous and the latter to discrete dependent variables. For the case of linear regression, it is assumed that the relationship among the variables is linear, so that the outcome of the dependent variable can be described as a sum of weighted observations from the explanatory variables [MPV01]. Additionally, a random error term is included in the linear function to account for the presence of noise in the data. In order to fit the linear function to a given data set, the least-square methods is used as an estimation technique [WMH60]. In contrast to linear regression, logistic regression allows the explanatory variables to be discrete. For this purpose, linear regression is based on a binomial regression function which models the probabilities of categorical outcomes [Men02]. Instead of a linear function, the relationship is modelled as a logistic function, which has a characteristic S-shape. As a closed-form solution is not available to find the required coefficients of the function, an iterative approach, e.g. Newton's method, is used to fit the model from data observations [Men02].

In its most widely used form, regression analysis is employed to compare and correlate the values of different data variables of the same time instant [DS98]. The rationale behind this approach is to obtain a predictor which allows for forecasting the outcome of a variable that depends on related observations [MPV01, DS98]. However, reasoning over temporal shifts in the data to extrapolate the outcome of the variable at future time instants with unobserved input is not possible. For future time instants, observations of the dependent variables have not been obtained yet, which are however expected as input by the model to make a prediction. Therefore, we can conclude that regression analysis cannot effectively support the goals of this thesis in learning patterns of temporal context changes and predicting humans' time-depending context.

2.4.3. Classification

Classification is one of the most widely studied prediction problems in statistical learning. Given different data categories, the idea of classification is to predict the missing category of a new data item [WFH11]. This basic problem has a wide range of applications in the field of data mining. For instance, in online marketing, it can be used to predict whether a customer is interested in buying a certain product based on the customer's profile (e.g. the user's gender, age, or education) [Ama11]. While a range of different approaches exist to perform classification tasks , we discuss two of the most popular methods in the following: the Naive Bayes Classifier and Decision Trees.

The Naive Bayes Classifier adopts Bayes' theorem for the sake of classification [WFH11]. For this purpose, the probability of a data item belonging to a specific class is computed given different data features. In order to reduce the complexity of the prediction task, the data features are assumed to be conditionally independent and a separate probability is computed for each data feature which is then combined into a single estimate. The prediction is then a maximum likelihood estimate of all classes. This means that, given the probability of all candidate classes, the one with the highest probability is chosen as a prediction.

In contrast, Decision Trees [Qui86] take an information-theoretic approach to predict a target class from given input features. For this purpose, a tree structure is learned from training data, where the nodes of the tree represent conditions over the data features, while the leaves correspond to different classes. The tree structure is optimized based on an information gain criterion, so that the information entropy over increasing tree levels is minimized. This leads to very compact trees and thus allows to understand the specific conditions that are most explanatory for different class memberships. The tree can then be applied to a new data item by testing the item's features against the conditions encoded by the tree to predict a target class.

While these models have applications in many areas, they are devised for classical data mining scenarios in which prediction is regarded as a classification task [Nee12]. Classification is an effective tool for establishing a relationship between data observations and the outcomes of a class variable [WFH11]. However, the representation of temporal changes in the data to reason over future outcomes at arbitrary time horizons is not supported. In order to enable the prediction of human context changes, statistical models are needed which can capture sequential patterns to describe the temporal evolution of a user's context. Classification models such as the Naive Bayes Classifier or Decision Trees are however not devised for time series analysis.

2.4.4. Bayesian Networks

Bayesian Networks (BNs) are a popular graphical probabilistic model, which can be used to encode complex probabilistic relationships for the purpose of prediction [RN09].

These networks are often used to determine the likelihood of uncertain events within a given domain of discourse when some evidence about available observations is given. For instance, in medical science, it can be used to study the probability of having diseases (e.g. cancer) given different symptoms of a patient [Bas00]. At the same time, it allows for reasoning in the backward direction, e.g., to determine the probability of observing different symptoms of patients suffering from a particular disease.

Bayesian Networks are based on a graph model, where the nodes of the graph represent random variables, and the edges model directed dependencies among them [Pea85]. The structure of the graph is often defined by experts of the domain of discourse and reveals conditional independence relationships. The random variables in this graph which share no direct connections are considered to be conditionally independent. Conditionally independent variables can be eliminated from the joint probability distribution, thus significantly reducing the complexity of inference. This way, probabilistic inference becomes possible in domains with a large set of random variables. Due to its graphical representation, Bayesian networks can be understood very well by humans and allow for much more obvious interpretations compared to other inference models.

Dynamic Bayesian Networks (DBNs) are extensions of Bayesian Networks, which are capable of modeling variables and their temporal relationships [Mur02]. This means that random variables are considered whose outcome evolve over time in discrete time steps. Two different types of these variables are distinguished in DBNs, the so-called observable and hidden variables. Observable variables can be measured in the real world and denote a direct stimulus to the model. For instance, ambient temperature as measured by a weather station is an instance of an observable variable. In contrast, hidden variables denote an artificial construct of the model, representing data which is not directly measurable, but has a latent relationship to the observable variables. Distinguishing between observable and hidden variables has proven to be effective when dealing with erroneous data, e.g., in case of a biased temperature sensor, since the biased state can be smoothed by representing it both as observable and hidden variable.

Dynamic Bayesian network can be used to solve difference inference problems related to the hidden variables in the model. Filtering refers to the problem of identifying the current states of the hidden variables based on the observable input. Prediction is the problem of extrapolating the future states of the hidden variables over subsequent time steps. These inference tools can be applied to wide range of problems dealing with data uncertainty. In particular, due to the separation of observable and hidden states, Dynamic Bayesian Network are often employed for the purpose of context recognition to predict the correct context state (hidden state) based on sensor data (observable state) [MM07]. Since our context prediction model is situated above the context recognition layer (see Section 2.3), smoothing and error correction tools are performed as part of the context recognition process. As a consequence, there is no need to distinguish between hidden and observable variables in our context prediction approach. We rather collect and predict context data sensed by the context recognition

which provides us a time series of observable input. Therefore, DBNs do not match the nature of the problem underlying our work targeting the prediction of observable context data.

2.4.5. Markov Models

Markov models are classes of stochastic processes which model the temporal evolution of random variables [How71a]. The simplest model is a Markov chain, which assumes the random variables to be discrete and evolve in discrete steps. A key characteristic of a Markov chain is that the state transitions adhere to the Markov property. The Markov property states that the temporal evolution is memoryless, i.e., that the next state is independent of all previous states, given the current state. The state transitions can therefore be encoded in a compact way as a discrete probability distribution. This assumption makes Markov chains a simple, yet effective method for representing real-world processes. For instance, it can be used to build random walk models, which are often applied to explain the rationale behind processes of observed choices which result in sequences of dynamic state changes.

More complex stochastic process models have also been developed, which are based on the Markov property. Semi-Markov chains [How71b] are generalizations of classical Markov chains which include an explicit representation of the temporal state dynamics. More precisely, in addition to the state transition probabilities, they also model the state dwell time, which refers to the time spent in a state until the next state change occurs. This eliminates a restriction of Markov chains, which allow the state dwell times to be distributed according to a geometric distribution only.

Another model variant are Markov Decision Processes (MDPs), which augment Markov chains with a decision logic component [Bel57]. They have applications in various fields of artificial intelligence, where actions need to be taken that influence the state of a system. For instance, the problem of directing robots in a way such that they reach desired target locations can be represented as a Markov Decision Process. For Markov Decision Processes, the model of Markov chains is extended with a choice of actions that can be taken in each state and a function that reveals the reward of reaching certain goal states. The solution of a Markov Decision Process then yields the best actions to apply in a given state of the system, which can be used for supporting decision-making in state-based transition system.

The simple, yet effective stochastic properties of Markov models make them an ideal candidate to reason over discrete context changes which are affected by the uncertainty in user behaviour. Therefore, in this thesis we rely on Markov models as a statistical framework to deal with various challenges in predicting state-based context data. In the rest of this thesis, we will show how the distinct, but related statistical properties of Markov chains, Semi-Markov chains as well as Markov Decision Processes can be exploited to address different context prediction problems in the focus of our work.

2.5. Summary

In this chapter, we have presented the relevant background information to lay the foundation of this thesis.

First, it has been shown that context prediction has recently gained much attention in research as a promising way to design and enable new proactive applications. In particular, the development of powerful context recognition technologies and the widespread usage of sensor-enabled mobile devices has made it possible to gather context histories which give insight into behavioural patterns of users for the prediction of future context. In order to highlight the great potential of proactive systems, different application domains have been explored which have a specific interest in future context information. In these domains, knowledge about the future context is exploited in various ways to improve the user's experience with computer systems. Specifically, we have seen that context predictions can help to increase the relevance of context-dependent information delivered to mobile users (mobile advertising), maximize the comfort provided by smart appliances (e.g. smart houses), decrease the cost of operating a technical system (e.g. wireless networks), or optimize the effectiveness of algorithms for mobile information dissemination (opportunistic communication).

Further, we have studied a generic architecture of proactive computing systems to clarify the role of context prediction compared to other data processing techniques involved with recognizing context data. We have described this architecture as a model of several data processing layers, of which context prediction constitutes the most upper layer dealing with time series of discrete context states. Then, we have discussed several well-known methods that can be applied to address the problem of statistical data analysis. This discussion has revealed that there are a range of different approaches (e.g. Bayesian Networks or ARMA models) available from the field of statistics and artificial intelligence, which are, however, most useful in specific application contexts only. We have analysed the properties of these predictors in detail and identified Markov models to be the most promising statistical toolkit for the task of context prediction based on their simple, yet effective prediction model for discrete random variables whose outcome undergo changes in time.

Consequently, this chapter shows that there is a close relationship between the problem of context prediction, methods of statistical analysis and the design of proactive computer systems, which will be explored deeply throughout the rest of this thesis.

Part II.

Context Prediction Models

Chapter 3

System Architecture

In this chapter, we give an overview about the design of our context prediction system for learning and predicting context patterns in human behaviour. For this purpose, we present a system architecture which entails a set of generic components involved in a context prediction process. In the following two chapters, we will then discuss the concrete instances of our context prediction system which are designed to solve specific prediction problems based on the described architecture.

3.1. Requirements

The design of our context prediction system is based on a number of important requirements, which relate to the overall goal of enabling accurate and expressive forecasts of state-based context data (see Section 1.1). In the context of our prediction system, these goals can be translated into various aspects of our system model, including desired properties of the prediction model, the degree of automatic learning, and the provided query interface as described in the following.

Uncertainty Modeling. User behaviour in the real world is affected by a strong degree of uncertainty. When observing human context events (e.g. a change in the user's location), the occurrence of the same events in the future can only be expected with a certain chance. A context prediction system should measure the degree of uncertainty in observed context patterns to identify the most reliable forecasts and enable applications to understand the confidence in the predictions returned. In order to accomplish this, likelihood measures from statistics and probability theory can be used, which provide us a framework for drawing conclusions based on given evidence about a user's observed past behaviour.

Multi-dimensional Context. The set of available context information, e.g. a user's locations and activities, has increased with advancements in the field of context recog-

nition. With each additional context type, the description of a user's context can be augmented with more and varied kinds of information. A context prediction system should be flexible with respect to managing multi-dimensional context information. This requires our system to be based on a generic context model which allows for processing and analysing arbitrary types of user context. In particular, there should be no restrictions with respect to specific types of discrete context that can be integrated into the context prediction model.

Sequential Behavioural Patterns. In the real world, users often make decisions which are built on each other and occur together. For instances, when visiting a sports club, a user might frequently go to a bar afterwards with his friends. Consequently, user behaviour should be learned in terms of the sequential patterns which can be observed in a series of context occurrences. A prediction system should be able to discover such behavioural patterns and exploit them for prediction. This will allow the system to accurately identify forthcoming context changes based on the most recent observations of the current user behaviour.

Integration of Domain Knowledge. Beyond context data obtained from sensors, also other sources of knowledge can provide valuable input to a context prediction system. Specifically, domain knowledge which gives insight into how people typically behave in target domains is very informative (e.g. in form of workflow models). Since this information gives insight into higher-level constraints which cannot be derived from sensor readings, the context prediction system should be able to integrate this knowledge into a prediction model of the user's behaviour.

Time Representation. For proactive applications, time is an important source of context information to anticipate when particular needs occur while users execute their activities. A prediction system should therefore reflect time as a first order construct in the prediction model. Specifically, the prediction model should allow for reasoning over context patterns in user behaviour and their time of occurrence. This will enable applications to predict what kind of activities can be expected next within a given time window to trigger suitable adaptations.

Unsupervised Learning. In order to create unobtrusive and intelligent services, manual interventions of the user should be minimized as much as possible. In particular, no explicit input of the user should be required to make predictions possible. To achieve this, behavioural user habits have to be learned in an unsupervised manner based the user's interactions with his environment. Hence, the prediction system is required to process streams of observed context changes to discover patterns of typical user behaviour in a fully automatic way.

Well-defined Prediction Semantics. In order to interact with a context prediction system, proactive applications need a well-defined query interface. For the specification of the queries, a natural language explanation is insufficient, since statistical prediction

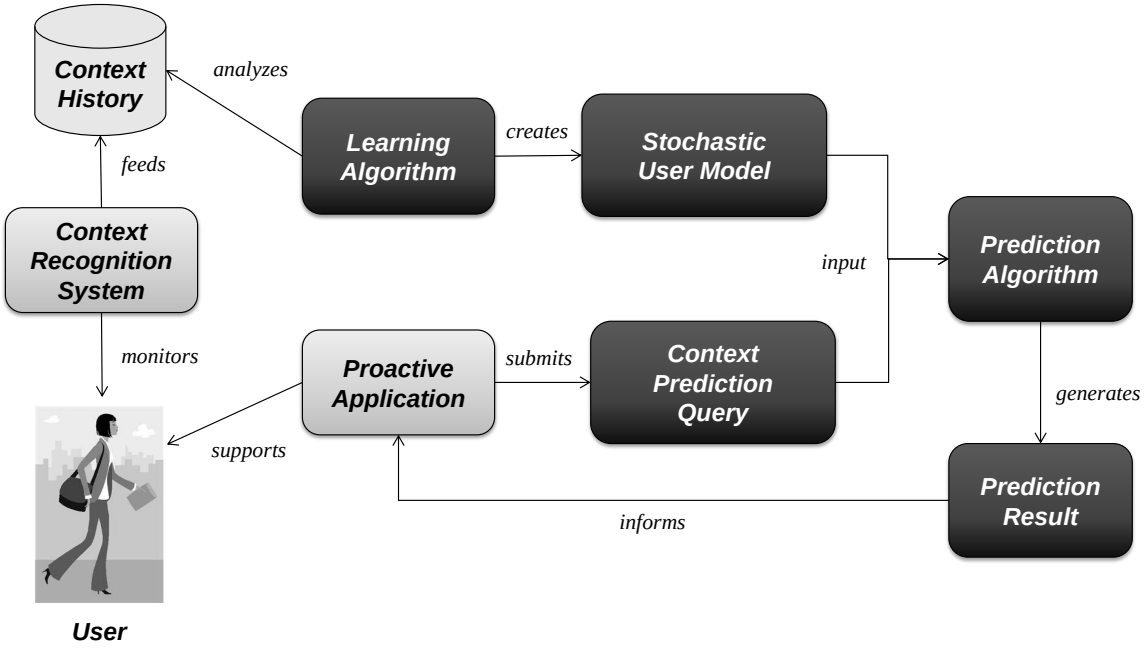


Figure 3.1.: Context prediction system architecture

models are complex and not easy to understand for non-expert users. Therefore, a clear formal semantics is required which describes the meaning of predictions in an unambiguous way. This will allow applications to make use of the full expressiveness of context predictions and apply them in the right context to achieve their goals.

3.2. System Components

We have designed a context prediction system which covers the requirements discussed above. The architecture of our context prediction system is illustrated in Figure 3.1. Since context prediction is a data-intensive task, it comprises various components, marked with dark boxes in the figure, which implement different aspects of data processing and analysis. In the following, we discuss the role of each individual component within our system in more detail.

3.2.1. Context History

The context history is a structured temporal database, which contains entries that characterize the past behaviour of users. Each entry refers to a time-stamped event that is related to a change in the user's context. Various types of context events may be recorded in the history that originate from a range of different context recognition

systems. For instance, enabled by location sensing systems that can track the movements of people, recorded context events may correspond to the user's visited locations. Similarly, the events in the context history may refer to information about a user's activities, e.g., modes of locomotion such as walking or sitting, which can be obtained from activity recognition systems [BI04]. Since all context events are associated with their occurrence time, context histories can be regarded as a time series of a user's context changes observed in the real world.

For our context prediction system, we assume that the context history stores discrete context data only. Most often, discrete context data is given as direct output from various context recognition systems. For instance, activity information as being detected by current systems is described with human-readable labels that are discrete by nature. Hence, this kind of information can be directly fed into the context history. In the same manner, many position systems such as the ones based on WiFi, RFID or Bluetooth allow for capturing symbolic locations to locate users [DR03]. Moreover, different techniques are described in literature to translate low-level data into discrete semantic context data (cf. Section 2.3). For instance, fine-granular user positions such as geometric coordinates that are returned by GPS receivers can be clustered to form places of spatial dimensions which are meaningful for users (e.g., home or office) [AS02]. Therefore, the context history is open for a range of different context types which can be included there as discrete context.

3.2.2. Stochastic User Model

The stochastic user model provides an abstract representation of the user's behaviour, which can be easily understood by computers. It is designed to give insight into meaningful patterns which are hidden in the observable data. Hence, in contrast to context histories, it does not encode isolated context events, but reveals relations of consecutive context changes and dependencies among different context types, e.g., location and activity changes. Since these patterns may be specific to different users, an effective prediction strategy requires an individual user model for every user whose behaviour shall be predicted.

In our context prediction system, the stochastic user model is based on a probabilistic representation. As real-world user behaviour is characterized by uncertainties and variances, a probabilistic model most closely reflects the choices people make. As a statistical framework, we use Markov models (cf. Section 2.4) due to their simplicity and effective nature for dealing with state-based variables. Our context prediction system exploits the properties of Markov models to map the user's behaviour to a state-based transition system, allowing for a) representing the context states in which a user resides b) analysing the user's transitioning behaviour among different context states and c) modeling the time a user spends in different context states. Since Markov

models can be applied to any kind of discrete context data, our stochastic user model is able to deal with information such as location, activity, etc. in a coherent manner.

3.2.3. Learning Algorithm

The degree of abstraction between context histories and the stochastic user model strongly differs: while context histories record detailed information about a user's past context events, the stochastic user model is supposed to store behavioural patterns only. The learning algorithm is intended to fill this gap by translating the raw events from the context history into the representation needed for the stochastic user model. For this purpose, learning algorithms are used which compute statistical summaries from sequences of past context events.

In our work, we use specific learning algorithms to train Markov models of a user's behaviour. This allows for learning the existence of a user's context states and the sequences of states typically visited by the user. With each observed context change, frequency statistics are updated that allow for deriving probabilities about the user's state transition dynamics. In order to account for deviations in the user's behaviour, the learning algorithm has to be executed periodically to update the stochastic user model with the most recent behaviour. Thus, outdated knowledge about past context changes that are no longer valid can be removed to avoid a negative impact on the accuracy of context predictions.

3.2.4. Context Prediction Query

In order to support proactive applications with knowledge about future context events, we provide a query interface to our prediction system. The query interface allows applications to choose among a set of different predictions which best suit the purpose for which the application is designed. Specifically, to enable proactive behaviours in a wide range of scenarios, we provide an expressive query interface which enables applications to pose queries with different operators and semantics.

We support different types of predictions in our context prediction system. From a abstract point of view, we distinguish between most likely and verification queries for future context, which both complement each other. Most likely queries ask for the context event to happen next with the highest probability. For instance, in case of mobility prediction, such a query could ask for the next visited location of a user, which would return the user's most likely destination. Most likely queries thus involve the selection of context events among a set of candidates with different occurrence probabilities. Verification queries, on the other hand, allow for predicting whether a certain pattern holds in the future, where the answer is of boolean nature. For instance, such a prediction could test whether a user will leave his current location soon. This

enables applications to explore whether a interesting situation might occur to trigger a particular service on behalf of the user. As an example, a proactive mobile recommender system could decide to deliver information about interesting events in a user's local surroundings if a user is predicted to stay at his current location for a sufficient amount of time. A detailed discussion of the prediction inference process and query semantics is subject to the next two chapters.

3.2.5. Prediction Algorithm

In order to answer a query for future context, a prediction algorithm is needed. The purpose of this algorithm is to explore possible behaviours of the user, and then to rate the probability with which this behaviour could hold. More precisely, a probabilistic inference process is executed where two sources of information are processed. On the one hand, the patterns contained in the stochastic user model are used as a search space of possible context changes. On the one hand, information about the most recently sensed context changes is used to condition the predictions on the current user behaviour. The user's current context is therefore integrated into the decision about what patterns are most likely to re-occur next.

Using Markov models as a statistical framework, the inference process in our system is conducted as a search algorithm in a state-based transition system. As part of the inference process, paths of state transitions are explored for which the probability is calculated to undergo sequential state changes. The inferred probabilities serve as a measure of confidence in the occurrence of a prediction. This gives proactive applications effective means at hand to judge whether a prediction is reliable or not. For application that require reliable predictions (e.g., smart homes), only forecasts with a high occurrence probability are incorporated. In contrast, for application where false predictions can be tolerated more often (e.g., recommender systems), also lower occurrence probabilities can be accepted to add more value to the provided services.

3.2.6. System Component Instances

As discussed in the last sections, we have designed our context prediction system using a generic component-based architecture. In order to address specific prediction problems in the focus of this thesis, dedicated instances of these components (e.g., prediction algorithms) are required. In the following, we list the component instances which have been developed as part of this work to increase the accuracy and expressiveness of context prediction systems. A detailed discussion of these components is subject to the next two chapters.

Context History:

- *Flow-enhanced History (cf. Section 4.2.1).* An extension of classical one-dimensional context history with information about the sequences of context changes and workflow activities.
- *Multi-dimensional Context History (cf. Section 5.3.2.1).* A generic representation of a multi-dimensional context history which may encompass context information from different sources and event types.

Stochastic User Model:

- *Markov Model (cf. Section 4.2.1).* A stochastic user model which represents the discrete context states and state transition probabilities associated with a user.
- *Semi-Markov Model (cf. Section 5.3.2).* A stochastic user model which extends classical Markov model with probability distributions about the time a user spends in different context states.

Learning Algorithm:

- *Bi-variate Markov Model Learning (cf. Section 4.3.2).* An algorithm to learn a bi-variate Markov Model which is able to reflect the correlation of two independent random context variables (workflow activities and classical context information).
- *Semi-Markov Model Learning (cf. Section 5.3.1).* An algorithm to learn a Semi-Markov Model which encompasses state transition probabilities as well as dwell time distributions associated with different user context states.

Context Prediction Query:

- *Most Likely Sequence of Context Changes (cf. Section 4.3.3).* Query for of the n most likely context changes which are to occur next.
- *Temporal Stochastic Logic (cf. Section 5.4).* Queries which are represented as temporal logic expression to describe time-dependent patterns of future context occurrences.

Prediction Algorithm:

- *Most Likely Sequential Path Mining (cf. Section 4.3.3).* A context prediction algorithm to answer queries for the most likely sequence of context changes.
- *Stochastic Model Checking (cf. Section 5.5).* A context prediction algorithm based on model checking techniques to verify the satisfaction of temporal logic expression for the occurrence of future context patterns.

Chapter 4

Improving Context Prediction Accuracy with Adaptable Pervasive Flows

4.1. Introduction

The design of proactive systems, based on knowledge of a user's future context, provides novel ways of supporting users more effectively in everyday tasks [PNF05]. In various application domains such as smart homes or pervasive health-care, context predictions can be exploited to minimize the amount of explicit input required by humans to control the surrounding computing environment [RBB+03]. However, due to the inherent uncertainty of real-world behaviour, the provision of services which act automatically on the users' behalf is a critical issue. In particular, as proactive services are based on assumptions about future context events, adaptations may also be experienced by users as confusing and disruptive if forecasts do not match their actual behaviour. One of the great challenges in context prediction research is therefore to design new prediction algorithms which can deliver accurate forecasts to minimize the disruptions noticed by users and guarantee a high level of unobtrusiveness.

Previous context prediction approaches rely on context histories as the only source of information for prediction [BD99, SKJH04, KM09, AAH+09]. Context histories provide insight into a user's context transitions, e.g., sequences of visited locations, which can be used to find patterns of re-occurring context changes. With this approach, context transitions which are frequently observed in the history can be reliably predicted. This way, often only the most popular context changes which dominate the context history are captured. Context changes which deviate from the popular patterns are perceived as abnormal behaviour. The weakness in the ability to understand different patterns of context changes negatively impacts the prediction accuracy. However, context change patterns share often a significant relation to the specific tasks and goals followed by users which serve as a motivation for their behaviour. In many application domains in the

focus of current research such as pervasive health-care, information about the regular activities performed by humans becomes available. Activity information can provide important evidence about the occurrence of upcoming context changes for enabling more predictions. For instance, consider a hospital scenario where workflow management systems are used to schedule the tasks of nurses. If we could access the information about the nurse's activity from the workflow management system, e.g. exposing that a specific patient needs to be visited next, we could use this information to forecast her next location more accurately. However, this requires to combine two different sources of information in a prediction model, enabling context predictions based on correlation patterns. As most of the current approaches deal with single-dimensional context only (e.g. a user's locations) [BD99, SKJH04, KM09, AAH+09], dependencies on a user's activities cannot be modelled and exploited for prediction. In order to overcome this problem, novel prediction algorithms are necessary which can take advantage and integrate information from activity-based models of human behaviour.

Based on this motivation, we propose a new prediction scheme leveraging the concept of *Adaptable Pervasive Flows* (APFs) that has been developed in the European research project ALLOW [FHH10]. APFs are context-aware workflows which are situated in the real world and expose the activities executed by human entities in various environments. In order to improve the prediction accuracy over current prediction systems, we propose an approach for exploiting flows as source of domain knowledge to provide context predictors with rich activity information. Since current context predictors are only devised for one-dimensional context data, we present a novel flow-based context predictor that is able to correlate observed context changes with the activities performed by humans. The predictor encodes this relation as a probabilistic state transition system that models the evolution of a user's behaviour based on a Markovian stochastic process. The state transition system reveals the conditional probabilities of a user's next context change given his current activity and last context state. For predicting the user's next context states, we then propose an efficient algorithm to traverse the transition system and compute paths of most likely context changes. In our evaluation, we show that our approach can significantly increase the prediction accuracy compared to state-of-the-art predictors, as conditional dependencies on the activities performed by humans are incorporated into the context prediction process.

The rest of this chapter is structured as follows. First, we introduce Adaptable Pervasive Flows in Section 4.2 as a model of context-aware human behaviour. We then present our flow-based context prediction scheme in Section 4.3, proposing new learning and prediction algorithms to exploit the flow-based knowledge. In Section 4.4 we perform an evaluation study and compare classical context predictors with the flow-based predictor we have developed. Subsequently, we discuss in Section 4.5 context prediction approaches from related work and show that no prior approach is able to correlate context changes with higher-level knowledge about a user's activities. Finally, we summarize and conclude this chapter in Section 4.6.

4.2. Adaptable Pervasive Flows

Workflows are inherent to human behaviour in the real world - either explicitly or implicitly. Explicitly, workflows can be found in domains such as hospitals or logistics, where humans follow common procedures in daily routines. In these domains, the activities carried out by employees often obey best practices or obligations as specified by an organisation's rules and regulations. Implicitly, even in less obvious daily situations, workflows are part of our everyday behaviour. For example, during sports, shopping or when engaging in leisure activities, people often behave in a structured way based on their routine behaviour. Consequently, workflows are widely available in various environments of daily life, making flow-based computing models an ideal choice for context-aware applications with a high intensity of human interaction.

Based on this motivation, Adaptable Pervasive Flows (also simply called *flows* hereafter) [HRKD08] have been proposed in the European research project ALLOW as a model for proactive applications. Flows are context-aware workflows situated in the real world that are based on machine-processable representations of the activities executed by humans. The overall idea behind this approach is that effective adaptations of applications become possible when activity information is directly integrated into the execution plan as a unified application model. In order to realize this idea, extensions to classical workflow technologies have been developed in ALLOW, which allow flows to adapt to changes in the activities of humans attached to them.

The basic flow-based computing system as described in [MPS+09, EFH+09] enables flows to react to changes in the current context of humans. In this chapter, we show how flows can also be exploited to enable proactive adaptations to future context changes. For this purpose, we propose a new flow-based context prediction scheme that can leverage on activity information as exposed by flows to increase the accuracy of context prediction. Before presenting our novel context predictor, we first describe a formal model of flows upon which our context prediction approach is based.

4.2.1. Flow Model

A *flow model* is a machine-processable description of a real-world process that involves human activities. Essentially, a flow model encodes the structure of activity-dependent processes under changing contextual conditions. This allows flows to model complex behaviours where the execution of human activities is subject to constraints in the execution order of these activities and dependencies on the execution context. For the definition of flows, formal modelling languages can be used which are based on a set of language-specific modelling constructs. In the following, we introduce a graph-based flow model which is supported by state-of-the-art workflow technologies and has a long tradition in workflow computing [BPE07]. Graph-based flow models are known

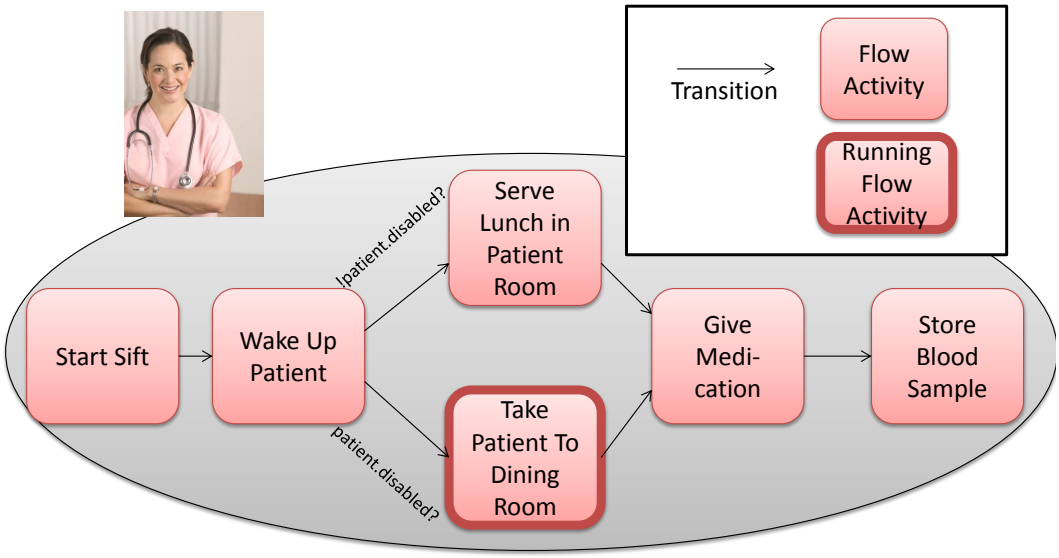


Figure 4.1.: Adaptable Pervasive Flow attached to a nurse

as sophisticated models with a clear execution semantics that allow for composing structures of various execution patterns in a generic and flexible way [Kop08].

Definition 1 *(Flow Model):* A Flow Model f is specified by a directed graph $(A, E, C, P(C), t)$, where

- A denotes the set of activities
- $E \subseteq A \times A$ defines a control flow
- C is the set of discrete user context
- $P(C)$ is the set of predicates over the user's context C
- $t : E \rightarrow P(C)$ associates each control link with a transition condition

The flow model describes a plan for the execution of activities based on a directed graph. The execution order is implied by the graph structure which introduces a control flow over the set of activities. For this purpose, each flow contains a start activity from which there is a path to any other activity in the flow. An activity path in the flow consists of sequences of activities which are linked through the graph edges, i.e., the path is defined by pairs $(a_i, a_{i+1}) \in E$ of an activity a_i and its direct successor a_{i+1}. In case an activity has more than one activity successor, the flow has a branched structure and models alternative paths that can be executed. In order to decide for one alternative, conditions are attached to the edges connecting an activity with its successor. The conditions are evaluated as predicates over the context $c \in C$ of a human whose activities are described in the flow.

As an example, consider the flow attached to a nurse in a health care scenario shown in Figure 4.1. The flow models various activities that a nurse needs to carry out for a patient treatment. For this purpose, the flow specifies health care activities such as "give medication" or "serve lunch". Also, the flow arranges the activities in a structure to reflect their temporal dependencies, e.g., a blood sample can only be stored after the medication has been given. As the required treatment activities may vary depending on aspects such as legal health care regulations, the activities to be executed are linked to conditions which are part of the nurse's context. For example, depending on the health conditions of the specific patient she is caring for, it may be decided where to serve the patient's lunch.

4.2.2. Flow Instance

Flow models serve as templates for executing flows at run-time. The run-time representation of a flow is referred to as *flow instance*. While flow models define a set of possible behaviours, flow instances keep track of the progress of the flow and expose the current state of the flow execution. Formally, we define a flow state as follows.

Definition 2 *(Flow State):* The state of a flow f at time t is formally specified as $s_t(f)$, and reveals the activity $a \in A$ which is currently performed in the flow.

The state of a flow instance is synchronized with the real-world environment using automatically collected context information in order to display which activity is under execution. To achieve this, flows are coupled with context recognition technologies to monitor a user's actions that are executed in parallel to a flow [HWR11]. The flow state evolves with the observed context and actions based on the following rules:

- When the flow instance is created, the state is set to the first activity described in the flow. Every flow has a unique activity which has no predecessor that serves as the flow's start activity.
- The start of a new activity terminates the previous activity and causes the flow's state to be updated. The state then changes from the previous activity to its successor. This signals a change in the execution of the current activity.
- Context conditions associated with the transitions are monitored to decide about the completion of an activity. Such a condition evaluates the current context of the user executing the flow. As long as the condition is not fulfilled, the current activity remains valid. As soon as the condition is satisfied, the current activity is terminated and the successor activity is started. Context conditions may evaluate various sources of information such as manual user feedback or the actions sensed by an activity recognition system.

- For activities with more than one successor, the next activity to be executed is found via a matching context condition. The context conditions of all transitions are evaluated to find an activity that is valid in the current execution context. The first activity for which the condition holds is then selected for execution.
- A flow is completed as soon as the last activity has been executed. The state of the flow is then retained as the activity which was executed last.

By following this model, flows are able to capture more expressive semantics than activity sensing technologies alone could provide. The reason is that flows are able to model sequential dependencies and integrate external contextual information (e.g. patients' health record) to describe a user's specific activity. For instance, while the user's action as sensed by activity recognition technologies yields "walking", the flow can assign the more expressive information "take a patient to dining room" to it. This allows flows to expose meaningful states about which activity a human is engaged in, as the interpretation can leverage on the wider application context.

In this work, we leverage on flows as a conceptual model of process-oriented context-aware applications. The implementation of the conceptual model with state-of-the-art workflow technologies has been discussed elsewhere [MPS+09, EFH+09, HWR11]. For the purpose of our work, flows are primarily interesting as a rich knowledge base to inform the process of context prediction with activity information. As a consequence, we do not focus on specific technological aspects in the following, but concentrate on a generic model for a context prediction system which is enabled by the described flow model.

4.3. Flow-Based Context Prediction

In this section, we introduce a new context predictor that is able to improve traditional context predictors with flow knowledge. Traditional context predictors are based context histories, which store information about the observed past context changes of a user. However, these context histories contain only one-dimensional context data, e.g. location changes, upon which a prediction is derived, but lack information about the user's higher-level activities which may help to improve the prediction.

In this following, we devise a new context prediction approach to combine both sources of knowledge – context histories *and* flows – to leverage the additional activity information present in flows. Key to this approach is a suitable representation of the relationship of flows and context to give insight into the context changes that frequently occur when executing specific activities. In order to realize this approach, we first propose a generic model of a classical context predictor and then propose an extension which allow us inject flow knowledge into these predictors to derive prediction which are conditionally dependent on a user's activity.

4.3.1. Context Prediction Models

As human behaviour cannot be described deterministically, the most common approaches for context prediction are based on stochastic principles. For this purpose, the occurrence of context (e.g. location) is regarded as a random variable X, which can be assigned values from a discrete set of context elements $C = \{c_1, c_2, .., c_n\}$. For the purpose of our work, we assume that each context $c_i \in C$ can be associated with a unique symbolic identifier. For example, in terms of geographic information, symbolic names such as "office" or "kitchen" can be used as location identifiers. A user's context history $H = (c_{t_1}, c_{t_2}, ..., c_{t_n})$ can then be represented as a sequence of context elements $c_{t_i} \in C$, which are ordered according to the time t_i of their occurrence.

The occurrence of sequential changes in context can be considered as a stochastic process χ that describes the evolution in user behaviour with distribution $P(X_1 = c_{t_1}, X_{t_2} = c_2, ..., X_{t_n} = c_n)$. The properties of such a stochastic process allow for making assumptions about the recurrence of a user's behaviour in the future which can be exploited for prediction. In the following, we introduce different predictors that leverage on these properties. First, we present an abstraction of Markov predictors, a class of stochastic models that are popular for dealing with processes of discrete states. Then, we introduce a new predictor which extends the classical Markov predictors with a bivariate state space to reflect conditional dependencies on a user's activities.

4.3.1.1. History Predictor

The most widely employed predictors [BD99, GC07, SKJH04] are based on discrete Markov processes. We refer to these approaches also as history predictors, since they are designed to consider a limited window of past context observations for prediction. Basically, there exist two different classes of history predictors - the fixed order *0(k)* Markov predictors and the predictors based on varying order Markov models. The order k of *0(k)* Markov predictors determines the number of last context changes $H(k) = (c_{n-k+1}, ..., c_n)$ from the history upon which the prediction is based. The simplest one is the first order Markov predictor with $k = 1$. In this case, the prediction of future context only depends on the last observation from the history. The restriction of a fixed order is relaxed by so-called Markov models of varying orders. The most popular varying order Markov predictors are based on the data compression algorithm of Ziv and Lempel [ZL78]. These predictors adaptively decide on the number of past context elements which are used for prediction: If the prediction benefits from more historic information, a larger number of past context changes is considered. Otherwise, shorter windows of past context elements are employed.

In order to devise a novel context prediction scheme which is compatible with any history predictor, we propose a generic prediction model based on a probabilistic state transitions system. This definition captures the common nature of history predictors:

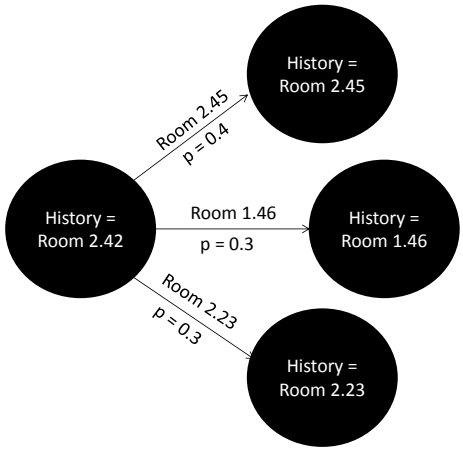


Figure 4.2.: Representation of a Markov model as state transition system

A state is a sequence of one or more past context elements upon which predictions are derived. We will then extend this generic model to combine it with knowledge about user activities. A very important consequence of this technique is that it allows us to apply our approach to the most important state-of-the-art context predictors.

Definition 3 *(History Predictor):* A history predictor $\hat{P}$ is specified by a probabilistic state transition system (S, C, δ, p), where

- C is the set of discrete context elements
- S denotes the set of history states
- $\delta : S \times C \rightarrow S$ denotes the transition function that describes possible context changes
- $p : S \times C \rightarrow [0, 1]$, indicates the probability for a specific context change with $\forall s \in S : \sum_{\forall c \in C} p(s, c) = 1$

The transition function specifies how history states are built from sequences of observed context changes. When observing a context $c \in C$ in state $s \in S$, the history state changes to $s' = \delta(s, c)$. Thus, the history state evolves with new context observations. The predictor encodes the probability of future context occurrences in transition probabilities: $p(s, c)$ indicates the probability for a change in context to $c \in C$ given the history state s.

In Figure 4.2, a history predictor is shown for a possible prediction scenario, where a nurse's next location shall be anticipated. The shown history predictor represents a first order Markov model *0(1)* as a state transition system. As the example shows, three possibilities exist for changing from "*Room*2.42" to other locations. However, all of these locations are almost equally probable. Therefore, a prediction will be only of

limited accuracy, since there is no clear preference to be made from the perspective of the history predictor when relying on past location traces only. In the following, we introduce a flow-based prediction scheme to alleviate this shortcoming.

4.3.1.2. Flow Predictor

In case of history predictors, only sequences of past context changes are used to compute a prediction. However, flows provide access to higher-level activities which can be exploited to further enrich context histories. By incorporating activity information, we can analyse if certain patterns of context changes listed in the history frequently occur in combination with certain user activities. This can reduce the prediction uncertainty, since important semantic knowledge becomes available on which upcoming context changes may depend.

Therefore, a novel prediction model is required which is able to reflect the evolution of context changes with respect to the specific activities executed by humans. Based on this model, those context changes can be predicted which are most likely happen for observed user activities. We address this problem by proposing a novel flow-based context predictor, which is able to express this relationship as a probabilistic state transition system.

Definition 4 *(Flow Predictor):* A flow predictor $\hat{P}_f$ is associated with a flow f and represents an extension of a history predictor $\hat{P}$. $\hat{P}_f$ is formally defined as probabilistic state transition system $(\hat{S}, C, \tau, p, f)$, where

- the states $\hat{S} = (S \times A)$ are the Cartesian product of the states of the history predictor $\hat{P}$ and activities of flow f
- C is the set of discrete context elements
- $\tau \subseteq \hat{S} \times C \times \hat{S}$ denotes the transition relation
- $p : \tau \rightarrow [0, 1]$ indicates the transition probability with $\forall s \in \hat{S} : \sum_{\forall c \in C, s' \in \hat{S} : (s,c,s') \in \tau} p(s, c, s') = 1$

States are now defined as tuples of flow activities and history states. Thus, we establish a relation among both. The flow activities in the flow predictor introduce a new differentiating criterion for history states. We can now find the *same* history state in different states of the flow predictor. Each of these history states may be associated with distinct transition probabilities. This enables accurate predictions tailored to the current user activity. In contrast, a history-based predictor represents each history state only once.

Figure 4.3 illustrates this difference for the prediction of the nurse's location. In contrast to the history predictor (see Figure 4.2), the flow predictor links locations to activities.

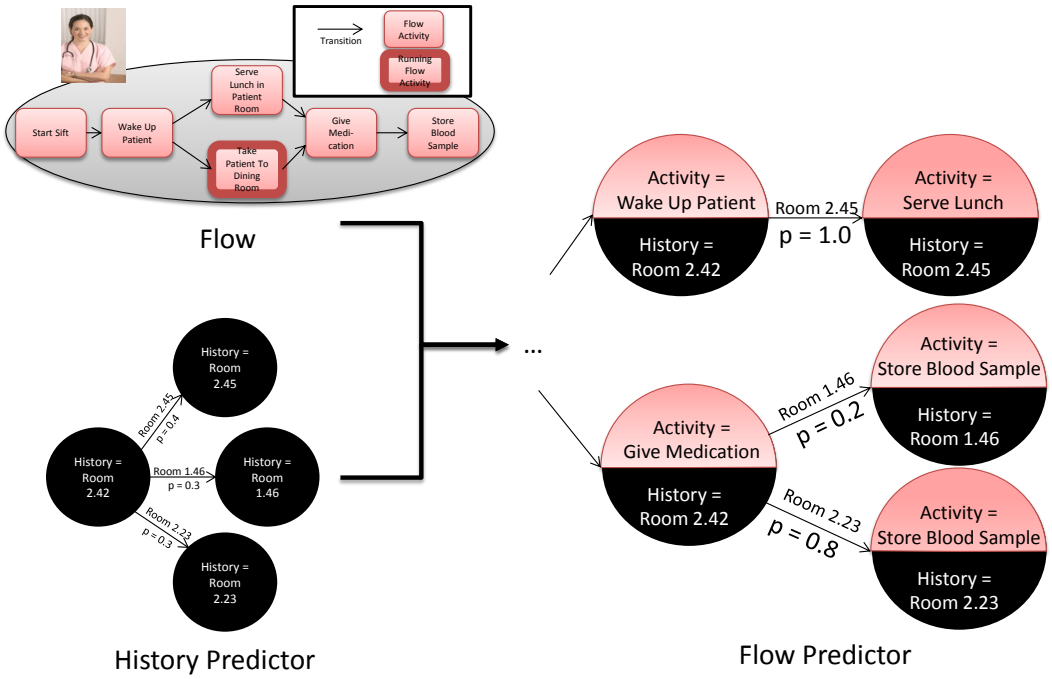


Figure 4.3.: Flow-based context predictor (right) combining flow activities (left top) and history states (left bottom)

Note that the state relating to "Room 2.42" has been split in two states, each being associated with a different activity ("Give Medication") and ("Wake Up Patient"). Using this activity information, the flow predictor can make a much more reliable prediction: When the nurse is executing the activity "Give Medication" in "Room 2.42", only two out of the three locations are likely to be visited, and the highest probability associated with an outgoing transition has increased over the history-based predictor. Moreover, if the nurse is executing "Wake Up Patient" in the same location "Room 2.42", then only one possibility remains for the next location ("Room 2.45"). Thus, by integrating activity information from flows, a single history state is split into multiple states such that for each of the following states the probability for its associated transitions may increase. Therefore, a higher accuracy can be expected for predictions returned by the flow predictor.

4.3.2. Learning Algorithm

In order to apply the prediction scheme to a specific scenario, an instance of our flow predictor is required where the underlying transition system reflects the user's actual behaviour. For this purpose, we propose an algorithm which is based on unsupervised learning to construct the transition system, i.e., no explicit input is required by the

Algorithm 1 Flow predictor: Learning algorithm

Require: $H_f =< e_1, \ldots, e_n >$
1: $\mathbb{B} \leftarrow \{\}, h \leftarrow \epsilon, a_{cur} \leftarrow \epsilon$
2: **for** $i = 1 \rightarrow n$ **do**
3: **if** $e_i \in A$ **then**
4: **if** $a_{cur} \neq \epsilon$ **then**
5: $\mathbb{B} \leftarrow \mathbb{B} \cup \{a_{cur}\}$
6: **end if**
7: $a_{cur} \leftarrow e_i$
8: **end if**
9: **if** $e_i \in C$ **then**
10: $c \leftarrow e_i$
11: $h' \leftarrow \delta(h, c)$
12: **if** $h \neq \epsilon$ **then**
13: UPDATE_PREDICTOR$(a_{curr}, h, c, h', \mathbb{B})$
14: **end if**
15: $h \leftarrow h'$
16: $\mathbb{B} \leftarrow \{\}$
17: **end if**
18: **end for**

user to complete the learning phase. This allows the learning algorithm to run as a background task unnoticed by the user for the provision of unobtrusive proactive applications.

Basic input to our algorithm is information about the user's executed activities and context changes that could be observed. Formally, this information is made available through the so-called flow-enhanced history H_f, which is defined as a sequence of events $H_f =< e_0, e_1, e_1, e_2, ..., e_n >$, where each event $e_i \in C \cup A$ can be either an activity ($e_i \in A$) or a context change ($e_i \in C$) ordered according to the time of occurrence. Note that the sequence of events can be arbitrary interleaved, i.e., there can be any number of consecutive activity executions while the context remains the same (and vice versa). The sequence of observed events contains important information about activity-dependent patterns of context changes which is learned by our algorithm.

The learning process is split in two parts, where Algorithm 1 shows the main loop of our algorithm. The algorithm sequentially iterates through the events from H_f (lines 2-18). Note that our predictor is trained in an on-line manner, i.e., the learning process is continued as soon as H_f grows and new information becomes available at run-time. Depending on whether the received event e_i corresponds to an activity a or a context c respectively, a specific action is chosen:

Algorithm 2 Flow predictor: Transition System Update

```
1: function UPDATE_PREDICTOR(a_curr, h, c, h', 𝔹)
2:     if (h', a_cur) ∉ Ŝ then
3:         Ŝ ← Ŝ ∪ {(h', a_cur)}
4:     end if
5:     if ((h, a_cur), c, (h', a_cur)) ∉ τ then
6:         τ ← τ ∪ {((h, a_cur), c, (h', a_cur))}
7:     end if
8:     count((h, a_cur), c, (h', a_cur)) ← count((h, a_cur), c, (h', a_cur)) + 1
9:     for all a_past ∈ 𝔹 do
10:        if (h', a_past) ∉ Ŝ then
11:            Ŝ ← Ŝ ∪ {(h', a_past)}
12:        end if
13:        if ((h, a_past), c, (h', a_cur)) ∉ τ then
14:            τ ← τ ∪ {((h, a_past), c, (h', a_cur))}
15:        end if
16:        count((h, a_past), c, (h', a_cur)) ← count((h, a_past), c, (h', a_cur)) + 1
17:    end for
18: end function
```

- In case the completion of an activity a is reported (lines 3-8), no update of the transition system is made yet. Since our predictor encodes the probability for a context change given a user's activity, we wait for the next context c to learn this relation. Since an arbitrary series of activity changes may happen in H_f before c may occur, we store the last activity in a buffer $\mathbb{B}$ (line 5). The rationale is that also for the previous activity a transition to the next context c should be learned. The buffer is then used as soon as the context c has changed to update the transition system as discussed next.
- For each received context update c (lines 9-16), we apply changes to the transition system. First, since a context change implies a new history state h', the previous state h is evolved to h' using c (line 11) (see also Section 4.3.1.1). Then, we request an update of the transition system for which we incorporate: the context change c, the old and the new history states h and h', the current activity a_{curr} as well as all pending activities stored in buffer $\mathbb{B}$ (line 13). Once the update has been completed, the buffer is emptied since the relation of all activities with c has been learned now (line 16).

The actual update of the transition system relates to two information aspects - the states of our predictor and the state transition probabilities. All required steps to perform the update are outlined in Algorithm 2. First, we verify whether the state transition system needs to be extended with new states. For this purpose, we test

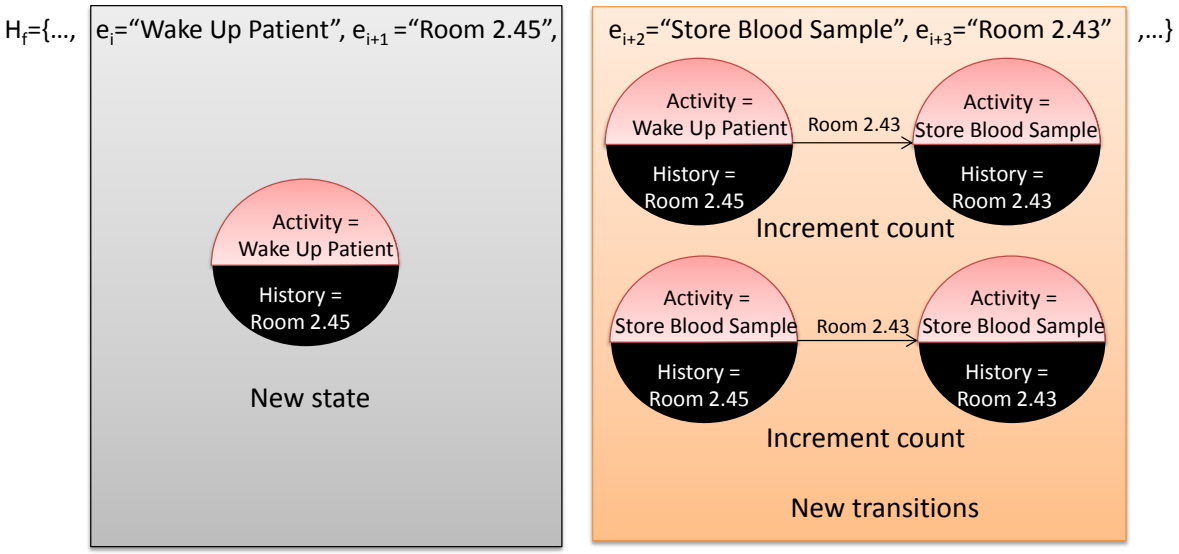


Figure 4.4.: Learning algorithm example for update of the transition system

whether the tuple (h', a_{cur}) has already been learned before (line 2). If not, it is added as a new flow predictor state to our transition system (line 3). The same is done for the transition $t = ((h, a_{cur}), c, (h', a_{cur}))$ (lines 5-7), since even though the source and target states of the transition might already exist, the transition may have never been taken before. Then, we update the frequency statistics which are the basis for computing the transition probabilities. For this purpose, every transition $t = ((h, a), c, (h', a')) \in \tau$ is associated with a counter, denoted as $count(t)$. If the transition is observed, its frequency counter is incremented (line 8). Finally, the context changes must also be associated with all activities buffered in $\mathbb{B}$. Consequently, the same steps (i.e. update to the predictor states and state transition probabilities) need to be performed for all activities contained in $\mathbb{B}$ (lines 9-17).

For example, suppose that H_f is defined as in Figure 4.4, where a sequence of possible events is shown which is processed by our algorithm. The states of our predictor result from combinations of subsequent events of activity and context information. As shown in the left part of the figure, a new predictor state s_1=("Room 2.45","Wake Up Patient") is learned in response to observing the events "Wake Up Patient" and "Room 2.45". Since the next event from H_f is an activity change ("Store Blood Sample"), the preceding activity ("Wake Up Patient") is buffered until the next context change occurs. Then, "Room 2.43" (context change) is observed, which causes a new state s_3=("Room 2.43","Store Blood Sample") to be inserted. Note that every activity should be linked to the next following context change. This encodes expected changes in the user's context given a particular activity. For this purpose, we insert a transition to state s_3 from both s_1 and the state s_2 = ("Room 2.45", "Wake Up Patient") associated with the previous activity. This is shown in the right part of the figure, where the two transitions

are illustrated.

The probability of a transition $t = ((h, a), c, (h', a')) \in \tau$ can be derived as a ratio over the observed frequencies associated with the transition counters:

$$p((h,a),c,(h',a')) = \frac{count((h,a),c,(h',a'))}{\sum_{c' \in C} \sum_{a'' \in A} count((h,a),c',(\delta(h,c'),a''))}$$

For the calculation of $p((h, a), c, (h', a'))$, we take all outgoing transitions from the state (h, a) into account. The probability is derived from the transition frequencies whenever a prediction has to be made. Note that the learning algorithm continually updates the transition counters for every transition which is repeatedly observed.

Due to the representation of activity information, there is an increased cost in storage associated with the flow predictor. The state space is of size $\mathcal{O}(|S| \cdot |A|)$, since activities are combined with the context elements. Consequently, also the encoding of transitions requires more space and has complexity $\mathcal{O}(|S| \cdot |C| \cdot |A|^2)$. However, for practical usage the cost will be affordable, as we assume a limited set of activities to be of interest and the real cost to be significantly below this worst case estimation, as activities are not observable at each context so that the state space is reduced.

4.3.3. Prediction Algorithms

In the following, we describe two algorithms which are able to exploit flow knowledge to compute context predictions. The novelty of our prediction algorithms is that they perform stochastic inference over a transition system which combines activity and context information. As we will see, this requires a novel approach where context changes are predicted in relation to the activities performed by humans. To this end, we describe algorithms to enable two cases of predictions - *short-term* and *long-term* context prediction as described in in the following.

4.3.3.1. Short-term Context Prediction

For short-term context prediction, we are only interested in predicting the next context change. Formally, let c_n denote the last context observed from H. The goal of the prediction is then to determine the context c_{n+1} that will most probably occur next. For instance, such a prediction could aim at the next location visited by a nurse, in order to prepare all electronic devices required for treatment procedures at this location already before a nurse arrives (e.g. for turning on the power of a device and completing the booting process).

In order to compute a short-term prediction, we explore the transition system for all successor contexts that are reachable over a single transition from the current state. However, we have to take into account that the same context c may occur in combination

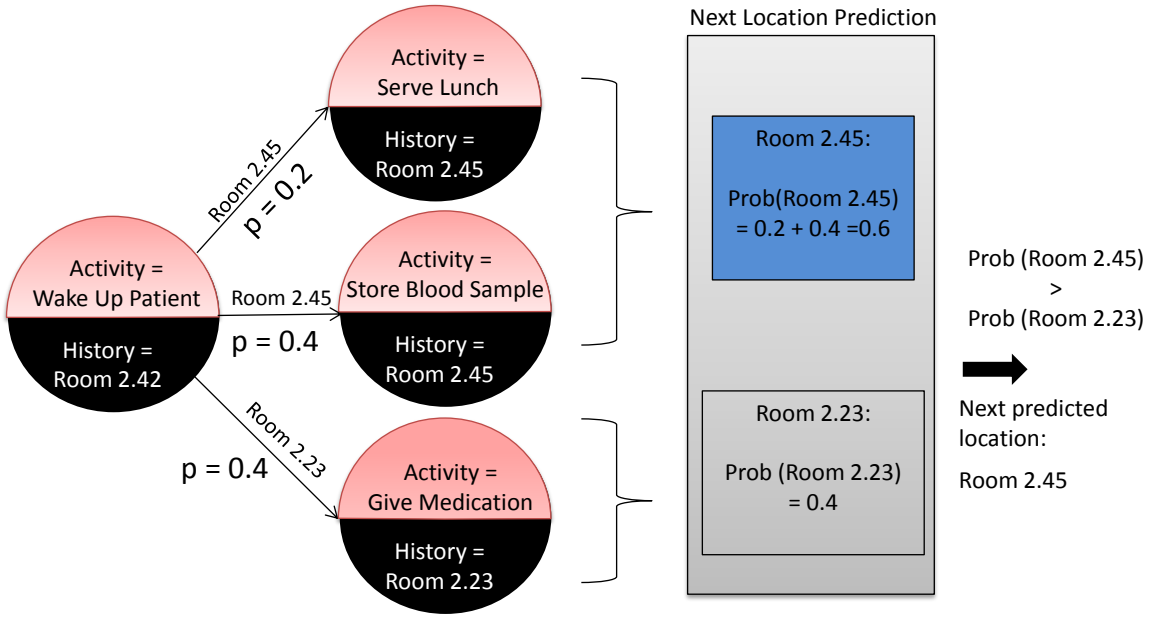


Figure 4.5.: Example of short-term context prediction

with different activities. This is because we have two-dimensional states in our transition system where a single context may be associated with different states as illustrated in Figure 4.5. As can be seen in the given example, the context "Room 2.45" is associated with two activities, while "Room 2.23" occurs only in combination with a single activity. Therefore, in order to identify the most likely context change amongst all, we have to compute the aggregated probability as the sum of all single transition probabilities associated with the same context c.

Algorithm 3 shows the steps involved in the calculation of the most probable next context. The starting point for short-term prediction is given by the current predictor state (h_{curr}, a_{curr}), from which the transition system is traversed. Then, we explore all outgoing transitions that are labelled with the same context c for different user activities a' (lines 3-8). In order to compute the probability for a context change, we accumulate the probabilities of all transitions linked to c (line 7). Finally, the context with maximum probability is selected and returned as a prediction (line 8). For the example shown in Figure 4.5, "Room 2.45" is predicted as the next location visit since the aggregated probability of 0.8 has the highest value among all probabilities.

In terms of computational overhead, the worst case time complexity is $\mathcal{O}(|A| \cdot |C|)$ since the next context may potentially occur in each of the flow activities. However, note than in practice the search space is much more restricted by the flow structure, where usually only a small subset of all activity-context combinations occur and thus a much smaller overhead can be expected.

Algorithm 3 Flow Predictor: Short-Term Context Prediction

1: $a_{curr} \leftarrow s_t(f)$ current flow activity
2: $h_{curr} \leftarrow$ current history state
3: **for all** $c \in C$ **do**
4: $\quad Prob(c) \leftarrow 0$
5: $\quad$ **for all** $a' \in A$ **do**
6: $\quad\quad h' = \delta(h, c)$
7: $\quad\quad Prob(c) \leftarrow Prob(c) + p((h_{curr}, a_{curr}, c, (h', a'))$
8: $\quad$ **end for**
9: **end for**
10: $c_{n+1} = \arg\max_{c \in C} Prob(c)$
11: **return** c_{n+1}

4.3.3.2. Long-term Context Prediction

For long-term prediction, we are interested in forecasting a sequence of successive context changes, conditioned on the user's flow of performed activities. More precisely, we attempt to identify the h context changes $c_{n+1}, c_{n+2}, ..., c_{n+h}$ which are expected to occur next. We also refer to this sequence as the most likely path, since it defines the path in the transition system with the highest probability over the next context changes. The length of the path is determined by parameter h, which is also referred to as prediction horizon. For instance, for $h = 3$ this could be the sequence of the next three locations visited by a nurse. This way, proactive applications are enabled that can gain insight into a user's behaviour which reaches far into the future.

However, in order to compute long-term predictions, the search space becomes more complex since the transition system needs to be explored over a number of subsequent state transitions. As the transition system exhibits a branched structure where a state may be connected to any other state, the number of possible paths increases exponentially with the given prediction horizon in the worst case. Hence, a naive search strategy which explores all paths of length h in the transition system is prohibitively expensive. Therefore, an approach is needed which is able to compute long-term predictions in an efficient manner in order to keep the search cost at a reasonable level.

For reducing the search complexity, our prediction algorithm is based on an optimised traversal strategy that avoids exploring all paths in the transition system. The optimisation takes advantage of the idea that certain paths can be ruled out at an early stage of the search procedure without harming the accuracy of the prediction. Given the goal of predicting the next most likely context changes, only paths need to be followed which denote viable candidates to become a most likely path. In contrast, if paths are encountered for which an alternative one exist with a higher occurrence probability, these paths can be pruned from the search. This decision can be taken before a path has been explored over its entire length as described in the following.

Algorithm 4 Flow Predictor: Long-Term Context Prediction

1: $a_{curr} \leftarrow s_t(f)$ current flow activity, $h_{curr} \leftarrow$ current history state
2: $Prob_0(h_{curr}, a_{curr}) \leftarrow 1$
3: $States(c) \leftarrow \{(h,a) \in \hat{S} | h = (c_{n-k+1}, \ldots, c_n) \wedge c_n = c\}$
4: $path \leftarrow \epsilon, global\_max \leftarrow 0, last\_context \leftarrow \epsilon$
5: **for** $i = 1 \rightarrow h$ **do**
6: **for all** $c' \in C$ **do**
7: $max\_prob \leftarrow 0$
8: **for all** $c \in C$ **do**
9: $tmp \leftarrow 0$
10: **for all** $(h,a) \in States(c)$ **do**
11: **for all** $(h',a') \in \hat{S}$ **do**
12: $tmp = tmp + Prob_{i-1}(h,a) \cdot p((h,a), c', (h',a'))$
13: **end for**
14: **end for**
15: **if** $tmp > max\_prob$ **then**
16: $max\_prob \leftarrow tmp$
17: $context\_change \leftarrow c$
18: **end if**
19: **end for**
20: **if** $(i > 1)$ **then**
21: $path_i(c') \leftarrow$ append $context\_change$
22: **end if**
23: **if** $(i < h)$ **then**
24: **for all** $(h,a) \in States(context\_change)$ **do**
25: **for all** $(h',a') \in \hat{S}$ **do**
26: $Prob_i(h',a') = Prob_i(h',a') + Prob_{i-1}(h,a) \cdot p((h,a), c', (h',a'))$
27: **end for**
28: **end for**
29: **else**
30: **if** $max\_prob > global\_max$ **then**
31: $global\_max \leftarrow max\_prob$
32: $last\_context \leftarrow c'$
33: **end if**
34: **end if**
35: **end for**
36: $i \leftarrow i + 1$
37: **end for**
38: $path \leftarrow path_i(last\_context)$
39: $path \leftarrow$ append end_state
40: **return** $path$

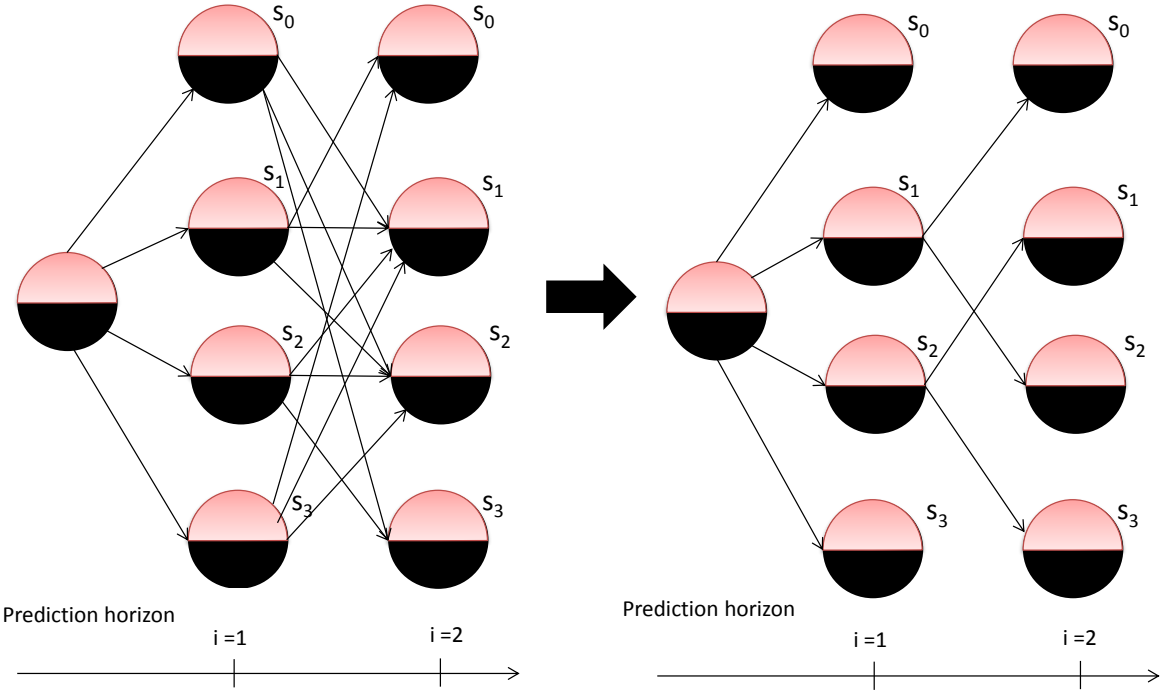


Figure 4.6.: Long-term context prediction exploiting a reduction of the search space

Formally, let $p_1 = (a_1, h_1), \ldots, (a_i, h_i)$ be a path of length i where (a_k, h_k) is a state that lies on the path. The probability that the path occurs is the product of the transition probabilities over consecutive states on the path. This probability is denoted as $Prob(p_1) = \Pi_{k=1}^{i-1} p((a_k, h_k), c_k, (a_{k+1}, h_{k+1}))$ with $h_{k+1} = \delta(h_k, c_k)$. Given two paths p_1 and $p_2 = (a'_1, h'_1), \ldots, (a_i, h_i)$ which terminate in the same state (a_i, h_i), it is possible to decide whether a most likely path of length $h > i$ is an extension of p_1 or p_2. This is because any such extension would start with the same state (a_i, h_i), so that the question of which extension would be more likely is determined solely by p_1 or p_2. In case $Prob(p_1) > Prob(p_2)$ only path p_1 is worth to be extended while p_2 can be dropped. Otherwise, p_1 is a candidate for a most likely path while p_2 can be safely omitted.

This principle is illustrated in Figure 4.6, where the left part shows the complete search space for the prediction of a most likely path. Searching through every state combination leads to a large number of different paths which needs to be considered. For increasing prediction horizons this becomes prohibitively expensive. The right part of the figure shows the result of applying our strategy for reducing the search space. Only a single path exists for each state in the transition system that is a candidate for a most likely path for a given path length. All other possible paths can be eliminated since they are of equal or less probability. By pruning the search space for candidates of most likely paths while exploring paths of increasing length, the search becomes much more efficient.

In order to implement this strategy, the algorithm follows a dynamic programming approach that incrementally calculates the most likely path of length h (prediction horizon) based on an iterative procedure. We leverage on dynamic programming as a technique to compose longer paths based on preceding calculations from shorter path lengths. In each iteration, paths are extended with one additional state until the prediction horizon is reached. The paths are then analysed for the probability that sequences of context elements occur, where paths with the same context elements but different activity information are merged. This approach is described in Algorithm 4 which consists of the following steps:

- The algorithm iterates through $i = 1, \ldots, h$ to compute path probabilities for increasing prediction horizons. In each iteration step, we determine the probability of a path comprising a context change c' at step i and context change c at preceding step $i - 1$ (line 12).
- In each iteration, the path associated with the highest probability is selected (lines 15-17), and the previous context change c is appended as next element to $path_i(c')$(line 21). Note that $path_i(c')$ is gradually extended and stores the sequence of $i - 1$ context changes from the previous iterations of the algorithm.
- In each iteration, we update the probabilities $Prob_i(h', a')$ to reach a context state (h', a') via a most likely path of length i (lines 24-28). For this purpose, we use the probabilities $Prob_{i-1}(h, a)$ that we inferred in the previous iteration for a path length of $i - 1$ and factor in the aggregated probabilities for extending the path with one further transition.
- As soon as the final prediction horizon h is reached, we select the most probable path of length h (lines 30 -33). The path is then extended with the final context change c' that marks the last element on the path (line 39), which is then returned as the prediction (line 40). Note that the prediction contains a path which stores the sequence of context changes $c_{n+1}, c_{n+2}, ..., c_{n+h}$ that are predicted to happen next.

Due to the dynamic programming approach, our prediction algorithm has a time complexity of $\mathcal{O}(h * (|A| \cdot |S|)^2)$. In contrast, the complexity of the naive exhaustive search algorithm has exponential overhead ($\mathcal{O}(h * (|A| \cdot |S|)^h)$), as all paths of length h need to be evaluated with this approach. Hence, the intelligent traversal strategy which we proposed, requiring states to be visited only once in each iteration, substantially improves the performance of the prediction algorithm. This guarantees a reasonable overhead of context predictions even for larger prediction horizons.

4.4. Evaluation

We have performed an extensive evaluation of our context prediction approach, analysing the extent to which flow-based knowledge can help to improve the resulting context

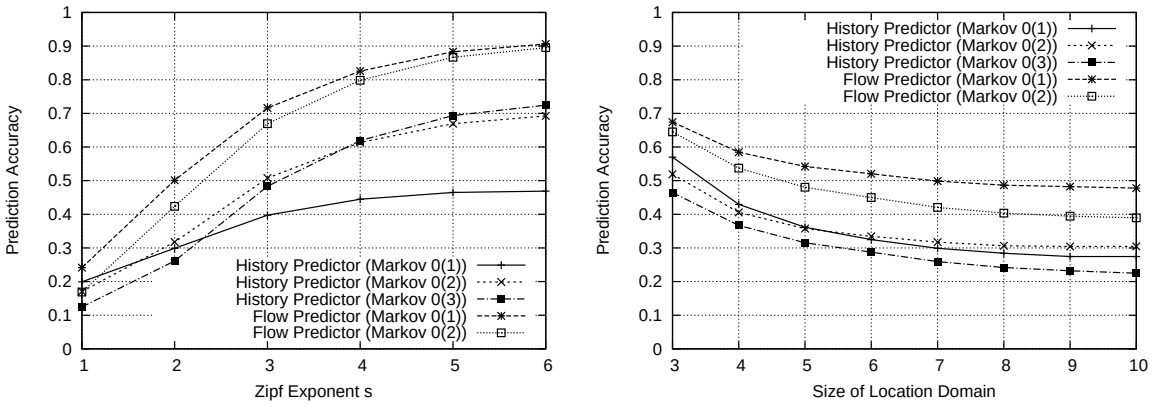
prediction accuracy. In the following, we first discuss our evaluation setup and then analyse the prediction accuracy that we gained for different evaluation scenarios.

4.4.1. Evaluation Setup

We have implemented a simulation environment in order to evaluate the effectiveness of flow knowledge for improving context predictions. This way, we could investigate the impact of different classes of user behaviour on the context prediction accuracy. In our evaluation, we contrasted our flow-based predictor with classical context prediction algorithms that rely on one-dimensional context histories. For the case of history predictors (see Section 4.3.1.1), we used the *0(k)* family of Markov predictors which condition predictions on the k last observed context changes. In contrast, our flow-based predictors are extensions of the Markov predictors which are able to incorporate activity information into the prediction process as discussed in Section 4.3.1.2. Although our approach is applicable to any form of discrete context, we focus in our evaluation on a location prediction scenario. In order to have fine-grained control over all aspects in the user's behaviour which can impact the context prediction accuracy, we use an activity-based mobility model that links a user's activities with location visits, as explained in the following.

In order to simulate a user's behaviour, we generated traces of a) the sequence of activities performed by users at b) the locations where these activities have been executed. For the task of a) we built flow models of different structure and size to consider a range of possible user behaviours. Specifically, we used two basic workflow patterns [BPE07], i.e., sequences and branches, to form flow models that allow for variable sequences of executed activities. For the task of b) we associated each flow activity with locations from the domain $L = \{l_1, l_2, ..., l_n\}$ where an activity could be executed. As reported in a large-scale empirical study of user behaviour [NSMP11], the relationship of user activities and location visits in the real world follows a power law distribution. In order to account for this, we derived location visits based on a Zipf distribution, a power law distribution for discrete data. The probability to visit the i-th location during an activity is given by $P(X = i) = \frac{i^{-s}}{\sum_{n=1}^{|L|} n^{-s}}$, and the choice of the exponent s allows us to vary the density of the visit probabilities at different locations. Based on this activity-based model of user mobility, we then generate sequences of activities (as obtained from paths in the flow model) and associated location visits (as obtained from the Zipf distribution). The sequence of activity and location information is then used to feed our context predictors and evaluate our context prediction approach.

We compare the different predictors based on an accuracy metrics which is defined as the ratio of the number of correct predictions over all predictions made. If a prediction is not possible due to the fact that the current context has not been learned before, we count it as incorrect prediction. Predictions are determined simultaneously to the learning phase, i.e., after each predictor update we compute a prediction and validate



(a) (b)

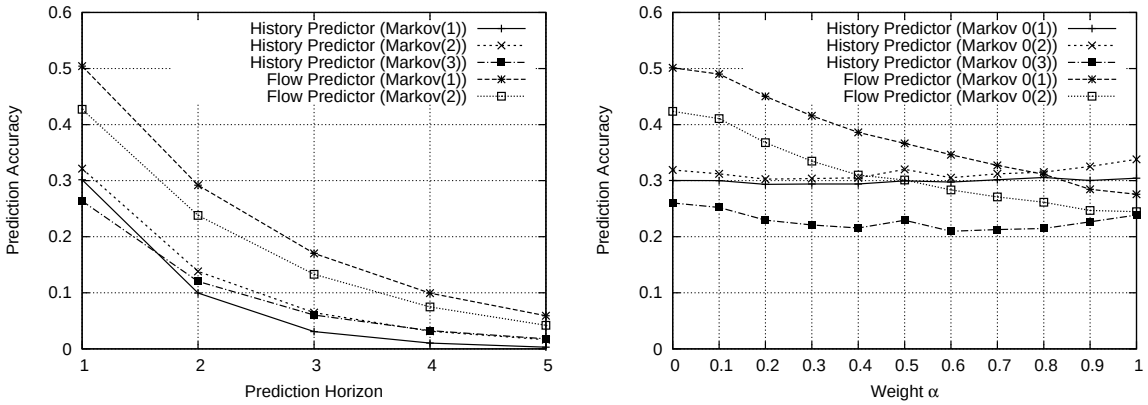
Figure 4.7.: Flow predictor vs. history predictor: short-term prediction

it. The results discussed in the following represent the average of 50000 predictions for each evaluation scenario.

4.4.2. Evaluation Results

First, we analyse the effects of varying Zipf exponents s on the accuracy of predicting a user's next visited location (short-term prediction). For this evaluation, we assume a location domain of $|L| = 7$. As Figure 4.8 a) shows, the prediction accuracy improves for all predictors with increasing exponents s. Since the location visit probabilities are more densely concentrated on single locations for higher values of s, all predictors are able to deduce a higher fraction of correct predictions. Among the history-based Markov predictors *0(k)*, higher orders of k are more effective for larger exponents s. This is because with more information from the user's context history, typical sequences of location visits can be learned which allow for predicting the next location more accurately. However, as our evaluation shows, the flow predictors, being able to capture activity-based patterns of a user's behaviour, significantly outperform the history-based predictors. More precisely, since the flow predictors can learn typical activity sequences and activity-to-location relations, unlikely locations can be ruled out and those locations visits can be predicted which are relevant to the user's activity. As our evaluation shows, the prediction accuracy is increased by 39% on average compared to the best history-based predictor given by Markov *0(2)*.

Figure 4.7 b) shows the short-term prediction accuracies for increasing sizes of the location domain given a fixed Zipf exponent $s = 2$. Due to the higher uncertainty associated with larger location domains, the absolute prediction accuracy is negatively affected for more distinct locations that can be visited. While this trends also holds for the flow-based predictors, they outperform the history-based predictors for all evaluated



(a) Long-term prediction (b) Hybrid prediction model

Figure 4.8.: Flow predictor vs. history predictor: further evaluations

location set sizes. For an increasing size of L, the relative improvement rises from 18% ($|L| = 3$) to 56% ($|L| = 10$) since the occurrence of more varied sequences of location visits can be better resolved when conditioning the prediction on the user's activities. We can further observe that the accuracy of the flow-enhanced predictors does not increase with higher orders k of the Markov model in our evaluation scenarios. As more context states have to be learned with higher orders, the learning process takes more time to converge so that prediction misses are caused. The flow predictor is designed to extract the required knowledge from the activity sequence information and the activity-to-location relations.

Figure 4.8 a) depicts the prediction accuracies for the case of long-term prediction, using a location set of size $|L| = 7$ and a Zipf exponent of $s = 2$. According to the definition of our accuracy metric, long-term predictions are only considered to be correct if the entire sequence of next visited locations holds. Hence, if only a single location visit does not hold that is part of the predicted location sequence, the prediction fails. For this reason, the prediction accuracies are exponentially decreasing with longer prediction horizons for both the history- and flow-based predictors as shown in the figure. While the absolute prediction accuracies fall to a low level, the relative improvement in accuracy of the flow-based predictor compared to the best history predictor increases from 57 % for horizon 1 to considerable 331 % for horizon 5. Consequently, longer-term predictions can benefit from our enhanced context prediction approach. This is because when each single prediction is improved due to the inclusion of activity information, the accuracy of the prediction of the entire sequence can significantly gain.

Finally, we analysed how our context prediction approach would perform if the dependency of the user's context changes on his activities is weakened. For this purpose, we introduce the parameter α that allows to vary between two modes of simulated user behaviour models, the activity-based model as discussed previously as well as a

history-based model. For the case of the history-based simulation model, we generate location changes from a $0(2)$ Markov source, i.e., the next location visit is chosen based on the 2 previous visited locations (and thus a user's activities are not relevant in the simulated behaviour). α can be chosen as a value in $[0, 1]$, indicating the portion of a user's behaviour which is generated by the Markov source and the portion $(1 - \alpha)$ which adheres to the activity-based model. In Figure 4.8 b), we can see the impact of increasing values of α on the prediction accuracy of the various predictors. As the history-based predictors are not able to capture activity information, their accuracy remains unaffected for any value of α. In contrast, the flow-based predictors show a monotonic decrease of the prediction accuracies for increasing values of α. As with higher values of α the relevance of activity information vanishes, the flow predictor gradually loses its capability to compute more reliable predictions. Nevertheless, still a higher prediction accuracy can be achieved for the majority of the cases even when the user activities become less important. Only for $\alpha \geq 0.8$ the history-based prediction approach slightly outperforms the flow-based predictors, since due to the correlation with user activities more context states are captured which causes the learning process to converge slower compared to the pure history-based predictors. However, the loss of prediction accuracy is marginal in this extreme case that can only occur when activities are not relevant to a user's mobility at all. Actually, it has been already empirically demonstrated that location visits depend on a user's activities [NSMP11]. As a conclusion, we can there observe that the flow predictor is able to effectively adapt to different scenarios and exploit activity information whenever it is available to improve the prediction accuracy.

4.5. Related Work

In the past, a number of different context predictors have been proposed to forecast discrete context data. While all these predictors rely on context histories as primary source of data, they differ in the algorithms used to perform the prediction. In the following, we compare the most well-known context predictors prevailing in current literature with the proposed flow-based context prediction scheme.

A general architecture for context prediction has been introduced by Mayrhofer [May04]. The architecture focuses on how to derive high-level user context (context classes such as "in a meeting") from low-level sensor data (e.g. location, noise, etc.) to populate context histories with meaningful data. As part of this work, different prediction approaches from continuous time series analysis (e.g. ARMA) as well as categorical prediction (e.g. Markov models) have been discussed. In order to overcome the limits in the achieved prediction accuracy, the author concludes that the inclusion of domain-specific knowledge would represent a promising solution. However, a concrete approach to accomplish this has not been proposed. In this chapter, we have addressed this issue

and showed how a context prediction can exploit the available activity information from flows to enhance the prediction accuracy.

Sigg argues that context prediction should directly take place on the level of low-level sensor data, since higher level context may introduce a loss of information that may negatively influence the prediction accuracy [Sig08]. For this purpose, an alignment predictor is proposed that can extrapolate a given time series of low-level numeric sensor data such as measurements of weather information. However, by focusing on low-level sensor data, meaningful changes in the user's context, e.g. a change in location from office to home, cannot be predicted which may involve several hours of measurements of low-level sensor data. Further, the proposed approach does not consider possible correlations among high-level context states that may give important evidence for the occurrence of future context. In contrast to the author, we argue that these correlations often appear on a higher level, such as the activities typically performed by users.

Furthermore, different algorithms have been proposed to allow for the prediction of specific categories of context. Most of the work in this area focuses on the prediction of user mobility from location histories [BD99, SKJH04, KM09, AAH+09]. The idea of these predictors is to find characteristic patterns in the sequences of past location visits that can be used as a basis for predicting a user's next location. For this purpose, often Markov models are used to represent a user's behaviour as a stochastic process [SKJH04, BD99]. Beyond pure location traces, however, these approaches do not consider any further information for prediction. In contrast to these approaches, we argue that higher-level information about the behaviour of humans can help to better understand and foresee a user's mobility patterns, and should be reflected in a prediction model. Therefore, by means of integrating user activities and traditional context such as location data, our work can be considered as an extension of these approaches.

Moreover, the prediction of low-level user activities such as key pressings to improve the interaction with user interfaces has been studied [HS07, DH98, GC07]. Similar to location prediction, future user activities are derived from histories of past sequences. Generally, the algorithms used are founded on different variants of Markov models. Thus, the prediction is determined based on the last seen symbols from the input sequence. In our approach, activities are the constituent parts of context-aware workflows that synchronize with the behaviour of humans in the real world. The difference is that we use the knowledge provided by the workflows to predict additional context that evolves with the activities performed by users, assuming an inherent relation between user activity and classical context such as location.

An efficient approach for the calculation of most likely paths in state-based transition systems is known from Hidden Markov Models (HMMs) [Rab89], where the Viterbi algorithm is used to discover paths of so-called hidden states for given observations. However, HMMs are based on a different probabilistic model that includes observation and transition probabilities. This model is not transferable to our problem, since there

is no such notion of hidden and observed states in the prediction of context data. In contrast, we deal with a transition system that consists of composite states defining a multi-dimensional search space. Therefore, in order to allow for long-terms predictions about a user's context changes, we have presented a new approach to compute most likely paths of states in a two-dimensional transition system.

The idea of exploiting knowledge about a domain of discourse to increase the accuracy of inference processes has been studied in other domains than context prediction. Specifically, in syntactic pattern recognition, the pattern structure of a phenomena is formalized and exploited to guide the classification process. For instance, based on a model that describes typical sequences of body movements, it has been shown that patterns such as gestures [BI98] can be accurately recognized. In this work, we argue along a similar line of argumentation, proposing an novel context prediction approach which can benefit from a structured model of human behaviour that can be found in relevant fields of pervasive computing such as pervasive health-care. To this end, we use flows as a source of domain knowledge to inform context predictions with activity information and identify activity-dependent context changes of the user more accurately.

4.6. Summary

In this chapter, we have presented a new context prediction scheme that is able to provide classical predictors with domain-specific knowledge inherent to flow-oriented computing environments. The domain-specific knowledge arises from a model of human-centric applications that describes the activities of mobile users as context-aware workflows. In order to improve the accuracy of context predictions, we have proposed an enhanced context predictor that is able to learn and exploit the relationship of flow activities with context changes observed in the real world. Our predictor is based on a probabilistic state transition system which encodes this relationship as transitions among two-dimensional states which include both activity and context information. For context prediction, we have then presented a novel algorithm to traverse the state space of paths over activity-dependent context transitions in an efficient manner.

In our evaluation, we have shown that the inclusion of knowledge about user activities in the prediction model significantly improves the prediction accuracy, as classical predictors are limited by their agnostic view on the application domain. Consequently, we have solved the first fundamental question of this thesis and demonstrated how further sources of information can be exploited to design novel prediction algorithms which are able increase the accuracy of context predictions.

Chapter 5

Expressive Context Prediction using Stochastic Model Checking

5.1. Introduction

In this chapter, we address the second fundamental problem that we have identified for increasing the practical relevance of context prediction systems. We argue that not only prediction accuracy, by also query expressiveness in context prediction is a critical factor for the proliferation of proactive computing systems. With query expressiveness, we refer to the variety and range of possible predictions that can be processed to support applications with knowledge about user's future context. In existing research, the focus has been mainly on enabling accurate predictions [AAH+09], while context prediction expressiveness has only received little attention. However, different applications may have a broad range of interests in future context information which cannot be fulfilled with a single type of query. Therefore, a powerful query interface should be provided which is based on rich semantics and allows for more varied predictions.

Unfortunately, previous context prediction approaches [SKJH04, KM09, HS07, GC07, DH98] are based on simple statistical models, which only allow for predicting the next context change of a user, e.g. the user's next visited location. More sophisticated context predictions that incorporate temporal restrictions and allow for reasoning over more complex expressions of possible behaviours are not feasible with these approaches. In particular, existing context prediction system are not able to provide support for a) temporal query constraints (restricting the time of when a context change should occur) b) logic-based expressions (allowing for predicting logical relationships such that location x should be visited but not y) and c) clear formal semantics (unambiguously clarifying under which condition a prediction holds). The lack of suitable methods to support these features represents a major restriction of current context prediction systems in supporting more sophisticated scenarios.

In order to overcome this shortcoming, we present in the following PreCon [FHR11], a novel context prediction method offering expressive prediction power. PreCon applies well-known methods of stochastic model checking [BK08], traditionally used for the verification of formal systems, to the analysis and prediction of human context. While classical model checking relies on hand-crafted models of computer systems, we extend its scope to domains where human behaviour shall be studied, and therefore an a-priori system specification is lacking. In order render these tools useful for the prediction of human context, we combine classical model checking tools with methods of statistical learning to form a novel powerful context prediction system. In doing so, we enable support for time-dependent predictions based on real-time constraints for the occurrence of future context events (e.g. "Will the user be at location x within the next 10 minutes?"), expressive query semantics using temporal-logic expressions (e.g. "Will the user be executing activity y at location x within the next 10 minutes?") and a clear prediction semantics based on the rules of formal model checking. None of these aspects are implemented in prevailing context prediction approaches. PreCon is the first approach to explore the integrated usage of logic-based prediction semantics and statistical methods for the prediction of a user's context.

For the design and implementation of PreCon, we propose a number of extensions to our context prediction system that we have introduced in Chapter 4. In order to allow for more expressive context predictions, we use Semi-Markov Models, a generalization of classical Markov models, to encode a user's time-dependent behavioural aspects. Moreover, PreCon is built on temporal stochastic logics, a probabilistic derivative of classical temporal logics, to enable predictions which are based on expressive temporal properties on future context. For answering a prediction, PreCon reasons over the Semi-Markov Model and verifies with what probability a temporal-logic expression holds. As prior model checking systems have been devised to support off-line analysis only, we propose extensions required for context predictions related to the run-time state in human behaviour. To this end, we adapt existing model checking algorithms to incorporate the user's running dwell time behaviour into the calculation of these probabilities. In our evaluation, we measure the performance of our approach for a real-world health-care scenario based on well-known metrics from information retrieval (precision, recall, and F-score) and demonstrate that PreCon can effectively support proactive applications with expressive context predictions.

The rest of the chapter is organized as follows. First, a basic overview of our approach is given in Section 5.2. Then, we introduce Semi-Markov Models as a prediction model in Section 5.3. Subsequently, we describe in Section 5.4 our context prediction query language which is based on temporal stochastic logics. The prediction algorithms behind our system are discussed in Section 5.5. In Section 5.6, we perform an evaluation of our approach using a real-world case study from the health-care domain. Then, in Section 5.7 we discuss the related work from both the field of context prediction and model checking. Finally, we summarize and conclude this chapter in Section 5.8.

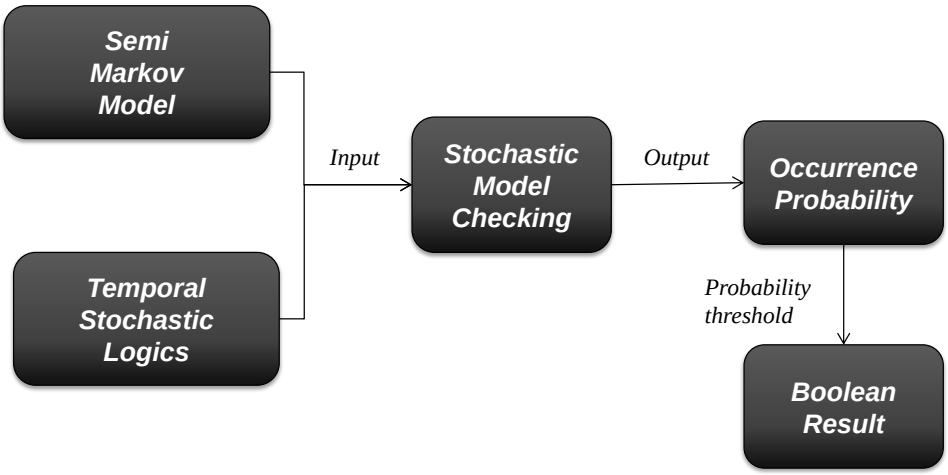


Figure 5.1.: Overview of the PreCon approach: context prediction using stochastic model checking

5.2. Overview of the PreCon Approach

We have designed PreCon as an extension of our context prediction system previously described in Chapter 4. In order to increase the expressive prediction power, PreCon introduces a number of new components which are shown in Figure 5.1. The key components of our extended prediction approach are a time-dependent stochastic user model, an expressive query semantics based on temporal-logical formalisms and novel context prediction algorithms given by stochastic model checking tools. In the following, we give an overview of these components that are specific to PreCon:

- **Semi-Markov Model**: A Semi-Markov Model (SMM) is an extension of classical Markov model capable of representing temporal aspects in a user's behaviour. Analogous to a Markov model, a SMM is a probabilistic state transition system that exposes the discrete states of a user's behaviour and the probabilities associated with state transitions. Furthermore, the temporal characteristics of state changes (the so-called dwell times) are modelled by the SMM. A dwell time is represented as a probability distribution over the time between two consecutive state changes and is associated with each state transition in the SMM. This is key to PreCon's concept of time-dependent predictions.
- **Temporal-stochastic Logics**: Applications can specify context prediction queries using temporal stochastic logic. Temporal stochastic logic is a derivate of classical temporal logic, which allows to query for temporal properties of systems which are characterized by uncertain behaviour. For the purpose of context prediction, we exploit this logic formalism to forecast whether a certain behaviour is expected to hold in the future. Due to the expressive power of temporal logic, the

query language provides well-defined semantics to express reachability properties (e.g. will the user arrive at a certain location) and invariant properties (e.g. will the user stay at a certain location) of a user's behaviour.

- **Stochastic Model Checking**: Model checking refers to an inference algorithm, which is used to verify formal properties of systems. For the sake of our problem, we exploit model checking as a context prediction framework. The input to the model checking process is the SMM representing a user's typical behaviour as well as the context prediction query expressed in temporal logics. A context prediction query is then evaluated on an SMM to calculate the occurrence probability with which the specified properties hold. A querying application can specify a probability threshold with which the resulting probability is compared, and a boolean result (true or false) is returned depending on the outcome. The querying application can take advantage of this knowledge to trigger proactive actions for enhancing the user's experience, e.g. adaptations of user interfaces and context-aware services.

As can be seen, the novelty of the PreCon approach lies in three different aspects - how user behaviour is represented, how predictions can be expressed and formalized, and how algorithms can be designed to infer expressive predictions. In the following sections, we will investigate each of these aspects in more detail.

5.3. Time-dependent Stochastic User Model

In the following, we give a precise formal definition of PreCon's time-dependent model of a user's behaviour. In this context, we also present our approach for learning such a model from observations recorded in real-world context traces.

5.3.1. Semi-Markov Model

We represent user behaviour as a Semi-Markov Model (SMM) [How71b]*. Markov models are a popular means for describing stochastic processes with discrete state spaces. In addition to classical Markov models, SMMs specify a so-called *state dwell time* – an arbitrary probability distribution that is associated with every state transition specifying the amount of time spent in a given state.

Definition 5 *(Semi-Markov Model):* A Semi-Markov Model (SMM) M is a 3-tuple defined as:

$$M = (S, p, h)$$

*Please note that the terms Semi-Markov Model and Semi-Markov Chain are often used interchangeable in literature

where S is the state space, $p : S \times S \rightarrow [0,1]$ with $\forall s \in S : \sum_{s' \in S} p(s,s') = 1$ is the transition probability function, and $h : (s,s',t) \mapsto [0,1]$ with $t \in \mathbb{R}^+$ represents the distribution of dwell times associated with a state transition $(s,s') \in S \times S$. For $h : (s,s',t)$, we will also write $h_{s,s'}(t)$ for brevity reasons.

The SMM allows us to describe a user's behaviour in the following manner: At each point in time, a user is in a state $s \in S$ that is identified by his current context (cf. Section 5.3.2.1). While the user acts in the real world, his context changes and the SMM moves to a new state $s' \in S$ representing the new context. s' is called the *successor state* of s, and s' is visited with a certain probability $p(s,s')$. Before leaving the current state s, s is active for a limited amount of time (the dwell time represented by $h_{s,s'}(t)$). During this time period the user's context does not change.

5.3.2. Learning Approach

In contrast to classical model checking, we do not expect a designer of the system to define the SMM underlying the real world behaviour. Instead, we apply a learning approach and derive the SMM from the observations of a context recognition system. In the following, we describe the basic elements of an SMM as well as the procedure of how to process context observations for learning.

5.3.2.1. User States

In order to represent a user's context, a state $s = (c_1, ..., c_n) \in S$ in the SMM is an n-dimensional vector of context information. Each component c_i of s is of a specific *context type* C_i with domain $Dom(C_i)$. E.g. c_i may be an integer value from $Dom(C_i) \subset \mathbb{N}^+$, and C_i may be the *ambient temperature* type. Other types could be *location* and *activity*, and the corresponding domains could be enumerations of possible activities and symbolic location identifiers respectively. Thus, the state space S is multi-dimensional and composed of all context types $C_1, ..., C_n$ known to the system with associated domains $Dom(C_1), ..., Dom(C_n)$. For example, the state $s = (\text{"}\textit{meeting room}\text{"}, \text{"}\textit{give presentation}\text{"}) \in S$ with $S = (Dom(Location) \times Dom(Activity))$ describes the fact that the user executes the activity *give presentation* in a location referred to as *meeting room*. Whenever a combination of context information $(c_1, ..., c_n)$ is detected that has not already been encountered for the specific user, a new user state $s = (c_1, ..., c_n)$ is added to the SMM.

5.3.2.2. Transition Probabilities

A concrete series of consecutive user states is represented as a stochastic process of random variables $X_1, X_2, X_3, ...$, where X_i refers to the state occupied after the i-th

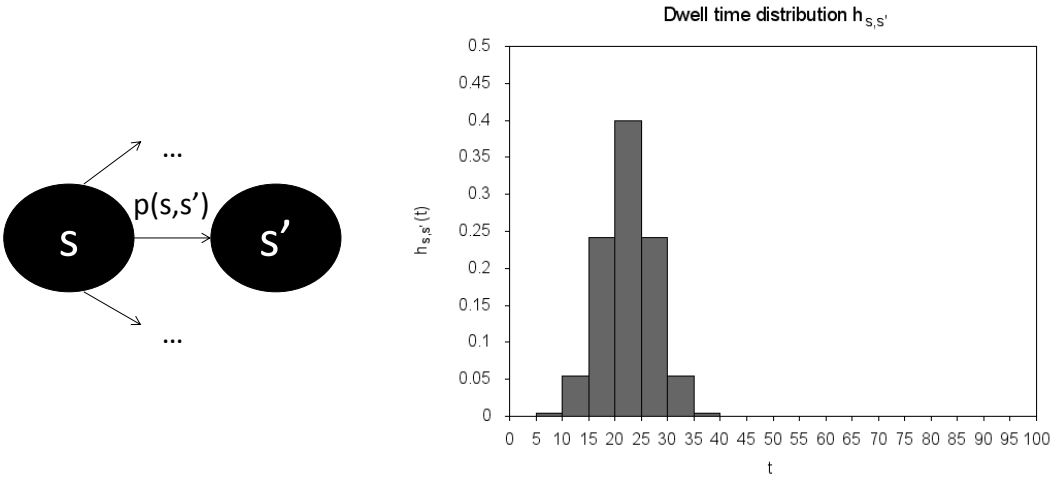


Figure 5.2.: Semi-Markov Model consisting of state transition probabilities (left) and dwell time distribution (right)

state transition. In order to learn the state transition probabilities, we assume the Markov property: The probability $p(s, s')$ for the state s' to be visited next only depends on the current state s, and is independent of all previous state changes. This assumption can be extended such that $p(s, s')$ depends on the k last visited states (k-order Markov models [SKJH04]) if needed, and PreCon operates on these more general k-order models. However, for simplicity, we assume $k = 1$ here. The math is essentially the same.

Assuming an underlying stationary probability distribution, the probabilities $p(s, s')$ can be estimated from the history of past state transitions: Let $w_{s,s'}$ be the *transition weight*, which denotes the number of transitions from s to s' as observed in the history. The transition probability $p(s, s')$ is defined as $p(s, s') = P(X_{n+1} = s' | X_n = s) = \frac{w_{s,s'}}{\sum_{s'' \in S} w_{s,s''}}$. Thus, the probability is the ratio of the number of observed state transitions from s to s' to the number of all observed transitions from s.

5.3.2.3. Dwell Time Distribution

The user's dwell time behaviour is modelled as a random variable D_n and denotes the period of time during which a user rests after the n-th state transition. For this purpose, we observe the time periods that pass between consecutive state transitions to learn the distribution of D_n. In order to account for specific dwell time behaviours in different states, we associate an individual distribution with every transitions from s to s', $s \neq s' \in S$ in the transition system. Formally, this distribution is represented as $h_{s,s'}(t) = P(D_n = t | X_{n+1} = s', X_n = s)$.

In order to limit the storage and computation overhead, we apply a discretization and divide time into intervals of equal size Δt, such that the i-th time interval is defined

as $I_i = [i \cdot \Delta t, (i+1) \cdot \Delta t)$. The distribution $h_{s,s'}$ can then be derived as follows: Let $w^i_{s,s'}$ be the number of transitions (s, s') that occurred in the interval I_i such that $w_{s,s'} = \sum_i w^i_{s,s'}$ is the total number of observed transitions (s, s'). Then, the probability for spending exactly time t in state s before leaving to successor state s' is calculated as

$$h_{s,s'} : t \mapsto \frac{w_{s,s'}^{\left\lfloor \frac{t}{\Delta t} \right\rfloor}}{\Delta t \cdot w_{s,s'}}. \tag{5.1}$$

In equation 5.1, we use Δt as a normalization factor to ensure that the sum of probabilities over all intervals is 1. This is required to establish $h_{s,s'}$ as a valid probability function. Figure 5.2 shows a example of such a dwell time distribution $h_{s,s'}$, where the time has been discretized into intervals of 5 time units. In this example, the distribution is concentrated on dwell times with a duration between 20 and 25 time units.

The cumulative distribution of $h_{s,s'}$ is given as $\int_0^b h_{s,s'}(t)\,dt$ and can be computed as the sum of of probabilities associated with intervals up to a desired upper time bound b. This expresses the probability for a state transition from s to s' to occur within the next b time units once s has been entered. The probability for a dwell time to lie in interval a and b can be derived from the cumulative distribution as $\int_a^b h_{s,s'}(t)\,dt = \int_0^b h_{s,s'}(t)\,dt - \int_0^a h_{s,s'}(t)\,dt$. As a and b may fall into discretization intervals, we interpolate the probability associated with the fraction of the covered intervals based on a linear function. The distribution $\int_a^b h_{s,s'}(t)\,dt$ is later used in Section 5.5 for the inference process to answer time-bounded queries.

5.4. Prediction Query Language

PreCon's prediction query language is based on *Continuous Stochastic Logic* (CSL) [BHHK03], a probabilistic derivative of branching-time temporal logics as known from classical model checking. CSL provides operators for verifying temporal properties of probabilistic state transition systems. We leverage on the expressive power of CSL to allow applications for composing temporal expression and submit them to PreCon for prediction. The operators are then evaluated on the learnt SMM to verify the specified properties on the user's possible future behaviour.

Consequently, context prediction strictly follows the formal semantics of model checking for state-based transition systems. According to this semantics, the state space S is traversed by going from one state to the next as the transitions among the states permit. The resulting series of visited states (called a *path*) models one possible temporal behaviour of the user. For a context prediction, PreCon starts at the state $s \in S$ the user currently occupies in the real world and evaluates the given query from there, possibly considering all possible paths starting at s (depending on the temporal operators in the query).

The concrete prediction semantics is based on the query language. For this purpose, let $p \in [0,1]$ be a probability threshold, let $\triangleleft \in \{\leq, \geq\}$ be a comparison operator, let $t \in \mathbb{R}^+$ be a time bound, and let $(C_i, c \in Dom(C_i))$ be a contextual value c of type C_i. Queries can be composed from CSL using the following grammar [BK08]:

A query is a temporal-logic formula Φ with

$$\Phi = true \mid (C_i, c) \mid \Phi \wedge \Phi \mid \neg\Phi \mid \mathbb{P}_{\triangleleft p}(\varphi),$$

where φ is a *path formula* defined as

$$\varphi = X^{\leq t}\Phi \mid F^{\leq t}\Phi \mid G^{\leq t}\Phi \mid \Phi_1 \, U^{\leq t} \Phi_2$$

The semantics of CSL is based on the classical operators provided by temporal logics to express *reachability properties* (using operators X and F) and *invariant properties* (using operators G and U) [BHHK03]. We leverage on these operators for the analysis of future user behaviour. X is the *Next* operator. It evaluates a condition Φ on all immediate successor states of the current user state s. Φ is expressed as a name-value pair (C_i, c) consisting of the name of a context type C_i (e.g. *location*) and a specific context value c (e.g. *office*). The query "Will the next location be the office?" can be expressed by applying the *Next* operator to $\Phi = (location, office)$, resulting in $X(location, office)$. F is the *Eventually* operator and can be used to verify if a condition Φ holds in any state reachable from s through paths in the SMM. G is the *Globally* operator and can be used to check if the condition Φ holds in every state on all paths starting in s. U is the *Until* operator and expresses that eventually Φ_2 must hold and Φ_1 must hold on all paths starting at the current state until Φ_2 holds.

Time is a first order construct of the prediction query language. All operators are associated with a time constraint t, defining an upper bound on the time, which may pass until the desired property holds. This enables applications to formulate time-dependent queries and set bounds on the time when predictions should occur.

The *raw* predictions are always probabilistic in nature when a query is evaluated. So the answer of the model checking algorithm is of the form "The user enters his office within the next 10 minutes with probability 0.74". A querying application, however, usually expects a *true* or *false* as an answer. Therefore, the calculated probabilities are compared to a probability threshold p, which is expressed in the subscript of a query formula ($\mathbb{P}_{\triangleleft p}(\varphi)$). The querying application specifies this probability and gets a boolean result depending on whether the outcome of the query evaluation exceeds the threshold or not. This is an abstraction implemented by the query interface provided to interested parties. If necessary, however, applications can also access the raw prediction results in case they require more complex threshold comparisons.

Query	Explanation
$\mathbb{P}_{\geq 0.8}(X^{\leq 10min}(location, office))$	Will the user go *next* to his office within 10 minutes with a probability of ≥ 0.8?
$\mathbb{P}_{\geq 1.0}\ (location, home) \wedge$ $\mathbb{P}_{\geq 0.8} F^{\leq 30min}(\neg(location, home))$	Is the user currently at *home* and will he *eventually* leave it within 30 minutes with a probability ≥ 0.8 ?
$\mathbb{P}_{\geq 0.6}\ G^{\leq 30min}\ (activity, walking)$	Will the user be *walking* within the next 30 minutes with a probability ≥ 0.6?
$\mathbb{P}_{\geq 0.2}\ ((location, stuttgart)\ U^{\leq 60min}$ $(location, home))$	Will the user be in Stuttgart with a probability ≥ 0.2 and then go *home* next within the next hour ?
$\mathbb{P}_{\geq 1.0}(activity, biking) \wedge \mathbb{P}_{\geq 0.8}(F^{\leq 60min}$ $(location, home) \wedge (activity, sitting))$	Is the user currently biking (anywhere) and will *eventually* relax ($activity = sitting$) at his home within the next hour with a probability ≥ 0.2 ?

Table 5.1.: Examples of temporal-logic prediction queries

In Table 5.1, we give some examples for behavioural properties which can be expressed as CSL formulas. The examples demonstrate the range of feasible context predictions PreCon offers, including queries with different semantics and context types.

The probability threshold p is an application-dependent value to influence the trade-off between *false positives* (queries that evaluate to *true* but prove to be false) and *false negatives* (queries that evaluate to false but actually become true in reality): A higher threshold reduces the number of false positives, but increases the number of false negatives. A lower threshold has the opposite effect. Consequently, the concrete threshold defines the ratio of false negative and false positives that the application is willing to accept. The choice for the threshold is dependent on the application semantics. For example, in scenarios where users want to be warned about service disruptions at train stations where they check-in next, user might be willing to accept false predictions to stay well-informed. Hence, such applications may tolerate a higher number of false positives rather than false negatives. On the contrary, a large number of false positives may negatively impact the satisfaction of a user. For instance, an advertising application which uses predictions of the locations visited by users to disseminate content should not obstruct the users with too many false predictions. In this case, a higher probability threshold would be beneficial to prevent the user from being spammed with irrelevant advertisements.

5.5. Model Checking Algorithms

Classical model checking algorithms assume static state transition systems, where the system is analysed at design-time and behavioural properties are only studied at state

entry times. In the case of human behaviour, the transition system that is subject to the verification is dynamic. In particular, the probability resulting from the evaluation of a query is depending on the time Δd that has passed since the current state s was entered. For instance, assume a situation where a worker has spent 8 hours at his current location *office*. The probability to finish work and leave the office soon is typically high in this situation considering typical work day schedules, and significantly different from the case where the work day has just started. Since model checking algorithms have not been designed to predict a user's real-world behaviour, both situations cannot be distinguished in classical model checking scenarios. More precisely, the existing methods do not consider the dwell time d as part of the model checking process.

In the following, we therefore extend the standard model checking approach to account for given values of Δd (referred to as the *running dwell time* in the following) by devising new ways of evaluating the temporal operators X and U. Since all other operators can be implicitly defined as expressions over X and U [BK08], extending the model checking algorithm for these two operators is sufficient. For example, the reachability property $F^{\leq t}\Phi$ can be transformed to the equivalent expression $(true\, U^{\leq t}\Phi)$. We therefore refer to X and U as the *basic operators* in the following. Arbitrarily complex temporal-logic formula can be evaluated in a bottom-up manner based on a tree representation [BK08] using only the basic operators.

Thus, evaluating a query requires two aspects:

1. We need to be able to determine whether a given state s satisfies a basic context constraint $\Phi = (C_j, c)$. The basic satisfaction relation is defined as $(s = (c_1, ...c_j, ..., c_n) \models \Phi) \Leftrightarrow c_j = c$.
2. We need the ability to calculate the probability of $X^{\leq t}\Phi_1$ and $\Phi_2 U^{\leq t}\Phi_1$ for some basic context constraints Φ_1, Φ_2. Intuitively speaking, this involves calculating the probabilities of reaching a state s with $s \models \Phi_1$ and of traveling a path where $s_i \models \Phi_2$ holds for every state s_i.

Our model checking problem can be solved by evaluating a satisfaction relation $\models$ for the path formula φ enclosed by the probabilistic operator $\mathbb{P}_{\triangleleft p}(\varphi)$ as follows:

$$(s, \Delta d) \models \mathbb{P}_{\triangleleft p}(\varphi) \Leftrightarrow P(s, \Delta d \models \varphi) \triangleleft p$$

In other words, the path formula φ is satisfied after Δd time units have passed in state s iff the probability $P(s, \Delta d \models \varphi)$ for the occurrence of φ satisfies the threshold condition $\triangleleft p$.

In the following, we will present the evaluation approach for the two basic operators in detail. Let i be the index of the last state transition that was observed, such that $X_i = s$ denotes the current state occupied by the user. Further, let $D_i = \Delta d$ denote the the running dwell time (cf. Section 5.3.2.3). As a common basis for the computations, we determine the probability for moving from state s to a successor state s' within time

t after a time Δd has passed since s was entered. This probability can be determined using the information carried in stochastic user model as follows:

$$P\left(X_{i+1} = s', D_i \leq \Delta d + t | X_i = s, D_i > \Delta d\right) \tag{5.2}$$

$$= \frac{P\left(X_{i+1} = s', \Delta d < D_i \leq \Delta d + t | X_i = s\right)}{\sum_{s' \in S} P\left(X_{i+1} = s', D_i > \Delta d | X_i = s\right)} \tag{5.3}$$

$$= \frac{p(s, s') \cdot \int_{\Delta d}^{\Delta d + t} h_{s,s'}(x)\, dx}{\sum_{s' \in S} p(s, s') \cdot \int_{\Delta d}^{\infty} h_{s,s'}(x)\, dx} \tag{5.4}$$

We use Bayes' rule to transform formula (5.2) into (5.3). As a result, the dwell time Δd is eliminated from the conditional part of the probability formula. Since the transformed formula matches the information stored in our SMM, it can be computed using the state transition probabilities and the dwell time distribution associated with the SMM (Equation 5.4). As the formula shows, the probability for a time-dependent state change from s to s' is composed of two parts: the transition probability $p(s, s')$ as well as the probability for the transition to occur within the next t time units as exposed by the dwell time distribution $h_{s,s'}$. Based on this probability, we can derive the predictions for the basic operators of CSL given by the next operator X and the until operator U as explained in the following subsections.

5.5.1. Next Operator

The predictions semantics of the Next operator X can be described as follows. Given a query $X^{\leq t}_{\Delta d}(\phi)$, the probability $P(X^{\leq t}_{\Delta d}(\phi))$ that the property ϕ holds in any state successor s' of the current state s is required. The subscript Δd in $X^{\leq t}_{\Delta d}(\phi)$ refers to the current dwell time, on which the prediction needs to be conditioned. Since $X^{\leq t}_{\Delta d}(\phi)$ is associated with a time constraint t, the probability of reaching s' within time $\leq t$ needs to be accounted for. Note that since our context prediction approach is generic in terms of the supported context types, ϕ may refer to any discrete context, e.g. a user's location in case of a location prediction scenario.

Lopez et al. have proposed in [LHK01] a model checking approach for SMMs which is able to process $X^{\leq t}(\phi)$ and compute the probability $P(X^{\leq t}(\phi))$. Their approach is sufficient for static systems where the evolution of the system over the period of time when a state is occupied is of no relevance. However, the current dwell time Δd which can be observed at the time a query is issued changes the probability of a prediction. In order to adopt model checking as a context prediction technique, we therefore extend the model checking approach described in [LHK01] to process $X^{\leq t}_{\Delta d}(\phi)$ as follows.

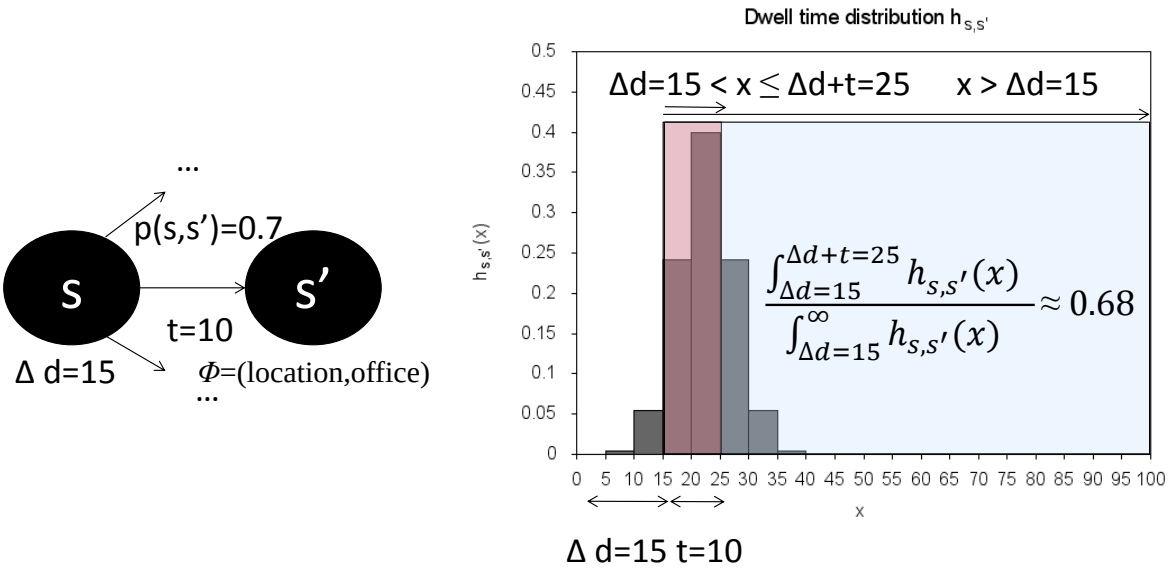


Figure 5.3.: Next Operator prediction inference based on state transition probabilities (left) and time-dependent transition probabilities (right)

$$P(X_{\Delta d}^{\leq t}(\phi)) \tag{5.5}$$

$$= \sum_{s' \in S \wedge s' \models \phi} P\left(X_{i+1} = s', D_i \leq \Delta d + t | X_i = s, D_i > \Delta d\right) \tag{5.6}$$

$$= \frac{\sum_{s' \in S \wedge s' \models \phi} p(s,s') \cdot \int_{\Delta d}^{\Delta d+t} h_{s,s'}(x)\, dx}{\sum_{s' \in S} p(s,s') \cdot \int_{\Delta d}^{\infty} h_{s,s'}(x)\, dx} \tag{5.7}$$

Equation 5.6 directly follows from the probabilistic formula given in [LHK01]. In contrast to the formula described by Lopez et al., we include the dwell time Δd as an additional evidence about the user' current behaviour in the conditional part of the formula. The formula then expresses the desired probability of reaching any state s' where ϕ holds (i.e. $s' \models \phi$) within the next t time units, given an observed dwell time of Δd. Note that, as ϕ may hold in more than one successor state s' due to the multi-dimensional state space S, this probability is computed as the sum over all time-dependent transition probabilities where ϕ can be reached in the next state.

We then transform this formula using Equation 5.4 to eliminate Δd as a condition event so that the formula can be computed using $p(s,s')$ and $h_{s,s'}$ from the SMM. This results in Equation 5.7, where the denominator is the probability of reaching arbitrary next states in greater than Δd time units, and the nominator limits this probability to

states where ϕ holds. The ratio represents the probability for a transition occurring within t time units given evidence that dwell time Δd has passed in s.

Let us consider the example shown in Figure 5.3, where $\phi = (location, office)$ is satisfied in s' and the transition probability is given as $p(s, s') = 0.7$. Further, we assume a current dwell time $\Delta d = 15$ and a time constraint $t = 10$. In order to resolve a prediction $P(X^{\leq t=10}_{\Delta d=15}(\phi))$, we determine $\int_{15}^{\infty} h_{s,s'}(x)\,dx = 0.95$ as the probability for a state transition occurring at any time $> \Delta d$. Similarity, we compute $\int_{15}^{15+10} h_{s,s'}(x)\,dx = 0.65$ as the probability of a state transition occurring at any time in between $\Delta d = 15$ and $\Delta d + t = 25$. Based on this information, we can infer $P\left(X^{\leq t=10}_{\Delta d=15}(\phi)\right) = p(s, s') \cdot \frac{\int_{15}^{25} h_{s,s'}(x)dx}{\int_{15}^{\infty} h_{s,s'}(x)dx} = 0.7 \cdot 0.6842 = 0.4789$ as the occurrence probability of ϕ to occur in any next state based on the given time constraints.

5.5.2. Until Operator

The Until operator U has more expressive prediction semantics, which makes a more complex model checking process necessary. In contrast to the X operator discussed in the previous section, the U operator predicts context changes which may occur at any state which can be reached within the given time constraints, not only at the immediate state successors. The exploration of the state space therefore extends to paths of subsequent state transitions. More precisely, according to the semantics of CSL [BK08], for a query $\Phi_1 U^{\leq t} \Phi_2$ all paths should be considered which end in states where Φ_2 is satisfied, while at all intermediate state Φ_1 holds.

In order to clarify this, consider a location prediction scenario, where the query $true\, U^{\leq t=10}_{\Delta d=15}(location, office)$ is given, i.e. we are interested in the probability of the user reaching his office within a time constraint of $t = 10$, given a current dwell time of $\Delta d = 15$. This scenario is illustrated in Figure 5.4, where the current state is s_1 and the context $\Phi_2 = (location, office)$ holds in state s_3. As can be seen, there is no direct transition between s_1 and s_3, and a path over a further state s_2 needs to be taken to reach s_3 from s_1. However, the time-dependent transition probability for transitions over several states is not directly encoded in the SMM. All we know is solely the transition probabilities $p(s_1, s_2)$ and $p(s_2, s_3)$, as well as the specific dwell time distributions h_{s_1,s_2} and h_{s_2,s_3} associated with each state transition. Consequently, an approach is required to combine the SMM information attached to single transition for calculating the probability for the user's behaviour over multiple state transitions. Note that while our example involves two subsequent transitions only, a generic solution is needed which can be applied to paths that may involve an arbitrary number of transitions.

In current literature, model checking algorithms exist which are able to reason over transient behaviours of Semi-Markov Models. In particular, the model checking approach of Lopez et al. [LHK01] is based on the following formula:

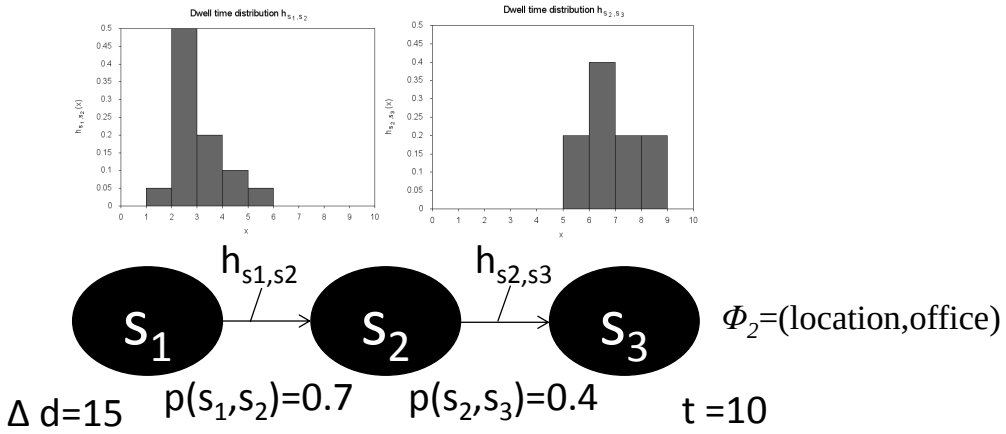


Figure 5.4.: Until Operator prediction inference requiring exploration of the transition system over paths of time-dependent state transitions

$$P(\Phi_1 U^{\leq t} \Phi_2) = F_a(s,t) \tag{5.8}$$

$$F_a(s,t) = \begin{cases} 1, \text{ if } s \models \Phi_2 \\ \sum_{s' \in S} \int_0^t p(s,s') \cdot h_{s,s'}(x) \cdot F_a(s', t-x) dx \\ \qquad\qquad , \text{ if } s \models \Phi_1 \wedge \neg \Phi_2 \\ 0, \text{ otherwise} \end{cases} \tag{5.9}$$

With this approach, $P(\Phi_1 U^{\leq t} \Phi_2)$ is computed using a stochastic inference function. The idea behind this approach is to consider the evolution of the transition system over paths of arbitrary state transitions starting from s that may happen within time bound t. As soon as a state s is entered where Φ_2 holds (upper case), a valid path has been found where the prediction holds, and the probability of reaching this state adds to $P(\Phi_1 U^{\leq t} \Phi_2)$. For discovering paths where $\Phi_1 \wedge \neg \Phi_2$ holds (middle case), i.e., all states on the path are valid but no terminating state has been encountered yet which satisfies the prediction. Therefore, the path is further expanded with a subsequent transition for which the time bound is updated. The idea is that if a transition from s to s' happens at time x, there is $t - x$ left for moving to a terminating state given an initial time bound t. Since we need to consider arbitrary times of when a transition may occur, the integral over the entire time interval is computed. In all other cases, the visited state is considered to be invalid since neither Φ_2 holds nor $\Phi_1 \wedge \neg \Phi_2$ (lower case).

PreCon's Until operator to support context prediction scenarios is defined as $U^{\leq t}_{\Delta d}$ and requires an extension to the above method. We additionally need to incorporate the

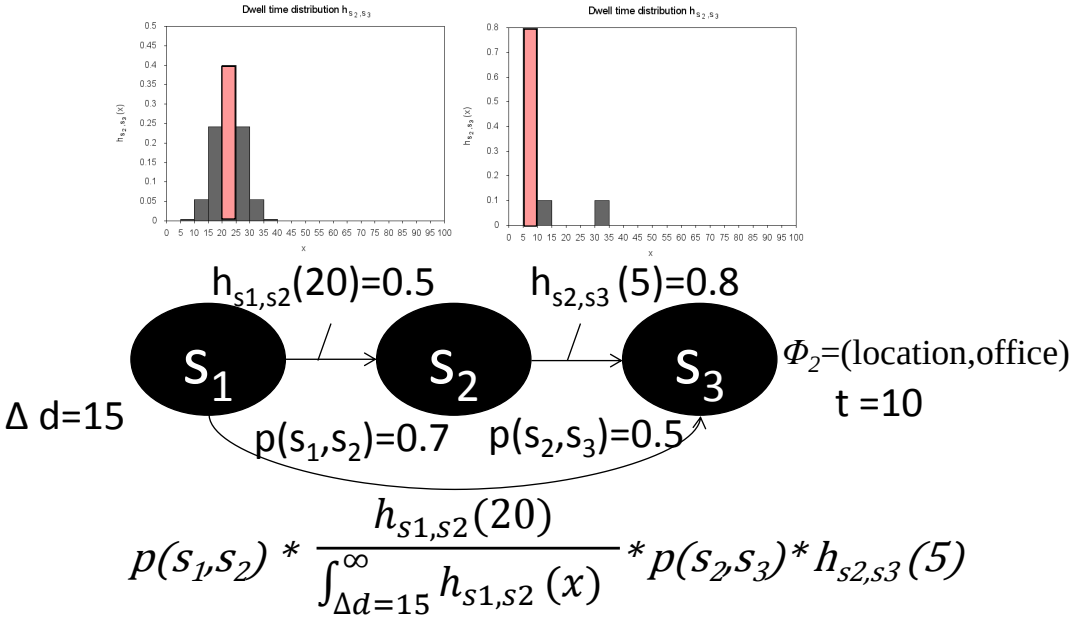


Figure 5.5.: Calculation of time-dependent transition probability for a path of two state transitions

dwell time Δd, expressed using a subscript in $U^{\leq t}_{\Delta d}$, as part of the inference process. This can be accomplished using this method:

$$P(\Phi_1 \, U^{\leq t}_{\Delta d} \Phi_2) = F_b(s, t, \Delta d) \tag{5.10}$$

$$F_b(s,t,\Delta d) = \begin{cases} 1, \text{ if } s \models \Phi_2 \\ \sum_{s' \in S} \int_{\Delta d}^{\Delta d + t} p(s,s') \\ \qquad \cdot \frac{h_{s,s'}(x)}{\int_{\Delta d}^{\infty} h_{s,s'}(x) dt} \cdot F_a(s, s', t-x) dx, \text{ if } s \models \Phi_1 \wedge \neg \Phi_2 \\ 0, \text{ otherwise} \end{cases} \tag{5.11}$$

(5.12)

Our prediction function F_b (Equation 5.9) introduces some required extensions to the original method F_a (5.11). First, the probability for the initial station transition needs to be conditioned on the dwell time Δd. Therefore, the integral borders are adapted so that Δd is set as the lower bound of when a transition can potentially happen. Second, the calculation of the state transition probabilities needs to be modified, since this probability is also dependent on Δd. In order to account for this, we compute the fraction $\frac{h_{s,s'}(x)}{\int_{\Delta d}^{\infty} h_{s,s'}}$ following the same rationale underlying Equation 5.4. In essence, since time Δd has already passed in the current state, we need to adjust the probability calculation for all possibilities which can happen from this point in time on. This means

that the probability for state transitions at any time prior to Δd needs to be eliminated from the calculation as done in Equation 5.9.

Figure 5.5 illustrates how to calculate time-depended transition probabilities on a path involving two state transitions. For the example, we assume that the first transition from s_1 to s_2 takes place at time $\Delta d + 5 = 20$, and the second transition from s_2 to s_3 is triggered at time $\Delta d + 10 = 25$. The probability for the first transition is then given as $p(s_1, s_2) \cdot \frac{h_{s_1,s_2}(20)}{\int_{\Delta d=15}^{\infty} h_{s_1,s_2}(x)dt}$, whereas the probability for the second transition can be determined as $p(s_2, s_3) \cdot h_{s_2,s_3}(5)$. Together this yields $p(s_1, s_2) \cdot \frac{h_{s_1,s_2}(20)}{\int_{\Delta d=15}^{\infty} h_{s_1,s_2}(x)dx} \cdot p(s_2, s_3) \cdot h_{s_2,s_3}(5) = 0.7 * \frac{0.4}{0.95} * 0.5 * 0.8 = 0.1179$ for the path from s_1 to s_3. Note that the probability decreases with increasing path length, since more uncertainty is involved in terms of user behaviour variances (due to the branched structure of the transition system and the spread in the distribution of the state dwell times). However, there may be multiple paths at various transition times which contribute altogether to the final occurrence probability as specified in F_b (Equation 5.9).

Note that F_b as developed in this thesis represents a generalization of F_a as proposed in [LHK01]. More precisely, in case of $\Delta d = 0$ where no dwell time has passed yet in a state, F_b falls back to F_a since $\int_{\Delta d=0}^{\infty} h_{s,s'}(t)\,dx = 1$ applies. Consequently, we have designed a generic approach that enables existing model checking systems to deal with run-time semantics. As we have done this for both the Next and the Until operator, PreCon enables the entire expressiveness of temporal logics in more dynamic scenarios where human behaviour needs to analysed and predicted at run-time.

5.6. Evaluation

We have evaluated PreCon using real-world context traces from a case study in a German geriatric nursing home. The nursing home is a health care facility where nurses care for elderly people suffering from dementia and other old-age diseases. As an application of an integrated context prediction system, PreCon has been employed to predict the future context of the nurses in order to optimize the tasks scheduled by an intelligent workflow management system. Incorporating context predictions into the scheduling decisions of workflow management systems is part of the European research project ALLOW [HRKD08].

In order to obtain the ground truth of a nurses' behaviour, the nurses were accompanied over the course of 25 days during 3-5 hours in the morning shift where they visited patients in different rooms to perform specific treatment activities (e.g., the patient morning hygiene). The observations of the nurses' behaviour were recorded in context traces which form the basis of our evaluation study. Each context trace consists of time-stamped entries of various context information including the activities performed by nurses, the locations of their visits and the ids of the visited patients. Thus, the

		Predicted	
		True	False
Real	True	TP	FN
	False	FP	TN

Table 5.2.: Classification matrix of prediction results

context traces define a time series of multi-dimensional context, where each entry denotes a discrete change of context associated with a nurse.

Given a context trace, PreCon learns a SMM to represent the behaviour of a nurse as a probabilistic state transition system with explicit dwell time characteristics (cf. Section 5.3). In our evaluation, we used variations of SMMs given three different state spaces comprising location, activity and patient information, i.e., $s \in Dom(Location), s \in Dom(Location) \times Dom(Activity)$ and $s \in Dom(Location) \times Dom(Activity) \times Dom(Patient)$. By varying the amount of information which is encompassed in a state, we can investigate the relevance of various context types for obtaining accurate predictions. For assessing the prediction performance of PreCon, we apply metrics from the area of information retrieval [MRS08]. These metrics are well suited to analyse prediction problems with binary outcomes as explained in the following.

5.6.1. Evaluation Metrics

PreCon offers a probability threshold to control the behaviour of the prediction system. Based on a given thresh configuration, we can distinguish between different classes of predicted and actual behaviour for analysing the performance of our system. If a prediction exceeds the probability threshold, then it will be delivered to the querying application. In this case, we count it as either *true positive* (TP) or *false positive* (FP), depending on whether the prediction matches the real-world context (TP) or turns out to be wrong (FP). In contrast, in case the prediction remains below the probability threshold, the querying application is not informed about the prediction. In that case, we distinguish between *true negative* (TN) or *false negative* (FN), depending on whether the suppressed prediction correctly did not occur (TN) or the actual behaviour occurred, but was not reported (FN). An overview of the different classes is given in the classification matrix shown in Table 5.2, were the rows represent the boolean outcomes of the real behaviour, and the columns show the outcomes of the predicted behaviour.

We count the occurrences of (TP), (FP), (TN), and (FN) over all predictions in order to assess the metrics *precision*, *recall* and *F-score*, which originate from the area of information retrieval [MRS08]. These metrics give insight into various performance

characteristics by building ratios over the frequency of the different classes of prediction results.

Precision is a measure of the exactness of the predictions, indicating the fraction of correct prediction out of all predictions returned to the application. Formally, it is defined as follows:

$$precision = \frac{TPs}{TPs + FPs} \tag{5.13}$$

Recall is a measure of the completeness of the predictions, and specifies the fraction of correct predictions out of all occurrences of the predicted behaviour (including those not seen by the application). Formally, it is defined as:

$$recall = \frac{TPs}{TPs + FNs} \tag{5.14}$$

The exact configuration of the prediction system is subject to a trade-off between precision and recall. Precision measures the accuracy of the predictions returned to the application. Therefore, configuring the prediction system with a high probability threshold such that only the most reliable predictions are returned will naturally lead to a high precision. In contrast, recall measures the extent to which predictions were missed in which applications have shown interest. Therefore, recall gains from making more risky predictions which are elicited for low probability thresholds. A good choice of the probability threshold guarantees an effective trade-off decision and is therefore important for the performance of proactive applications.

In order to measure this trade-off, F-score is a known method to combine precision and recall into a single metrics [MRS08]. Formally, F-score has the following definition:

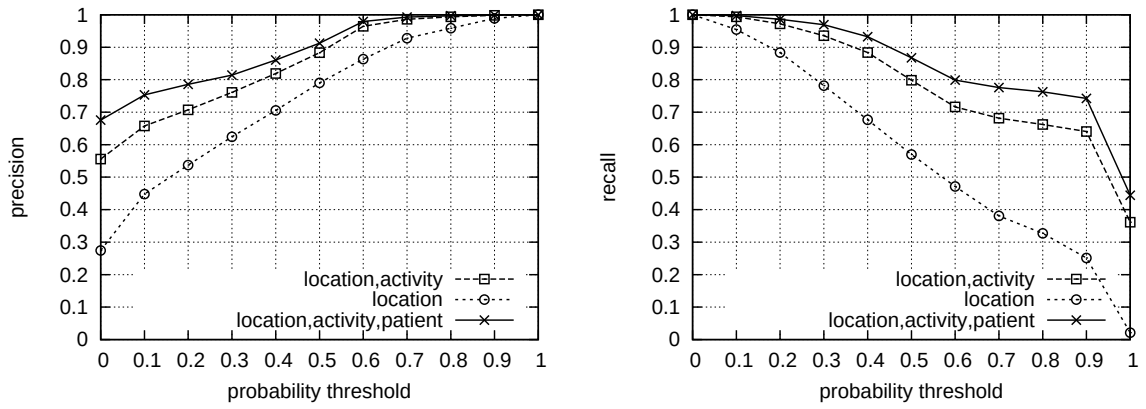
$$F\text{-}score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.15}$$

i.e. it is defined as the harmonic mean over precision and recall. This metric makes the consequences of the trade-off decision explicit and measurable. A high F-score indicates a good trade-off between precision and recall.

5.6.2. Evaluation Results

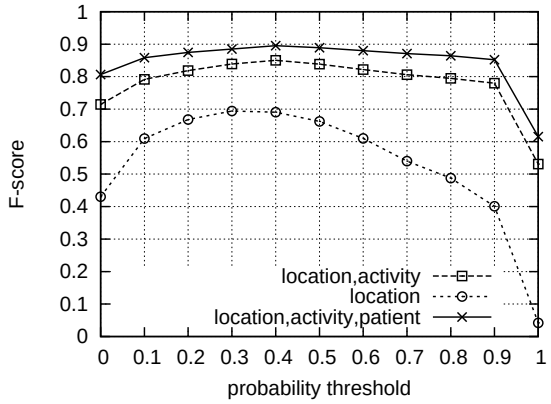
In our evaluation, we investigate how the performance of PreCon is influenced by a varying probability threshold for different queries. Using precision, recall and F-score as evaluation metrics, the goal is to recommend probability thresholds which should be chosen in order to achieve the best prediction results.

Based on the real-world data from the previously described health-care scenario, we have evaluated queries which involve the prediction of the future location of a nurse.



(a) precision

(b) recall

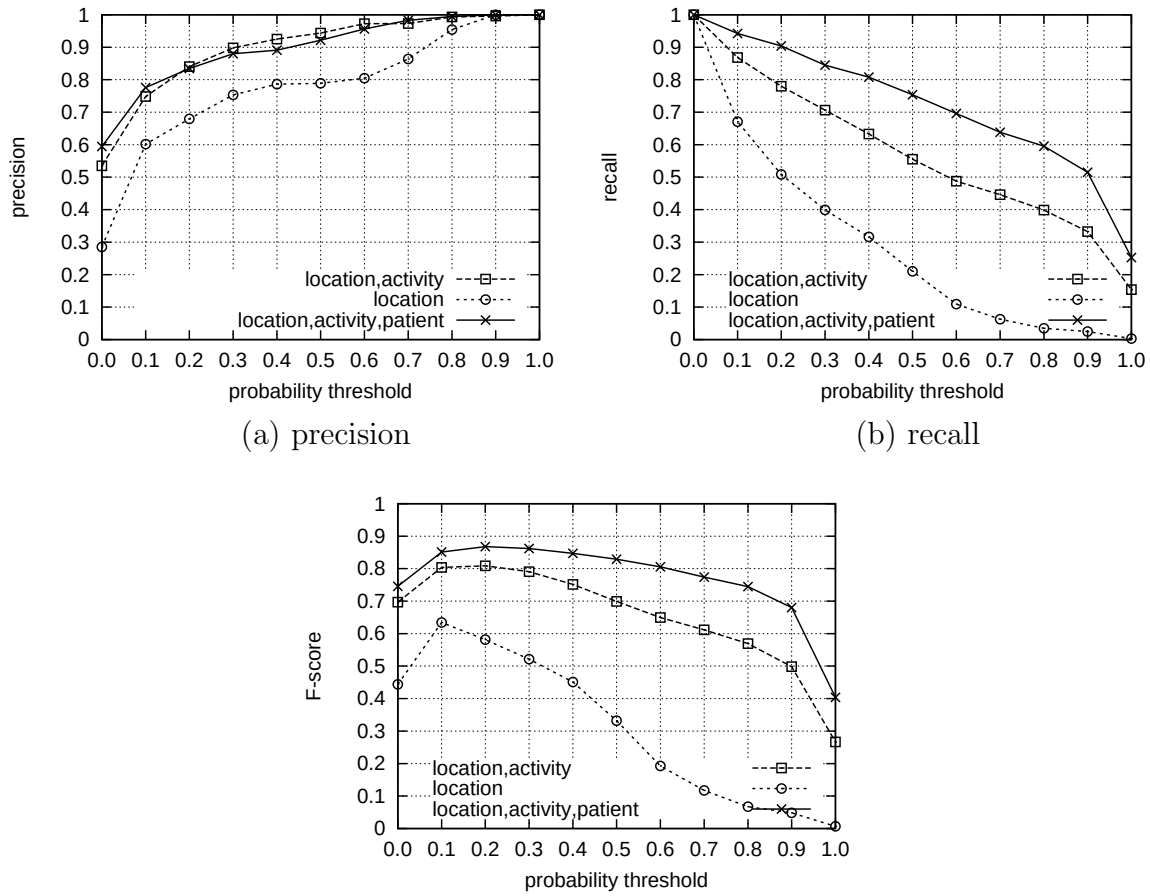


(c) F-score

Figure 5.6.: Prediction results for the Next operator: a) precision (top left) b) recall (top right) and c) F-Score (bottom)

For this, we defined two different queries $\mathbb{P}_{\lhd p}(X_{\widehat{f}}^{\leq t}(\varphi))$ and $\mathbb{P}_{\lhd p}(\ F^{\leq t}\varphi) = \mathbb{P}_{\lhd p}(\ true\, U^{\leq t}\varphi)$ using the basic temporal operators provided by our query language (cf. Section 5.4). Based on these queries, we verify whether a future location is reachable in the next state (using the *Next* operator $X_{\widehat{f}}^{\leq t}(\varphi)$) or at some state in the future (using the *Eventually* operator $(F^{\leq t}\varphi)$). The time constraint t associated with these queries is set to 10 minutes, and we defined instances of φ for each location in the nursing ward. We evaluated the queries repeatedly, i.e., predictions were computed upon a state change and periodically after $\Delta E = 10$ seconds have passed in a state. The results discussed in the following show the average of 2000 predictions. Depending on the given probability threshold, each prediction has been assigned to the different result classes as required by our metrics.

First, we study the results for the *Next* operator $X_{\widehat{f}}^{\leq t}(\varphi)$ based on the different metrics as shown in Figures 5.6 a) - c). As expected, the precision gains from an increase of the



(a) precision

(b) recall

(c) F-score

Figure 5.7.: Prediction results for the Until operator: a) precision (top left) b) recall (top right) and c) F-Score (bottom)

probability threshold as shown in Figure 5.6 a). The reason is that the number of FP decreases for higher thresholds because a larger portion of predictions with a low occurrence probability is discarded. The highest precision can be observed for predictions which incorporate all recorded context ($S = Dom(Location) \times Dom(Activity) \times Dom(Patient)$). In contrast, using only location information ($S = Dom(Location)$), the precision stays significantly below. Hence, the evaluation results show that additional context is relevant to discriminate user states such that more accurate predictions can be expected. Figure 5.6 b) shows the recall as a function of the probability threshold. The results are reciprocal to the precision results: For an increasing probability threshold the number of FN increases, as more and more predictions are discarded that actually occur but do not reach the required occurrence probability. This result illustrates the trade-off between precision and recall for different values of the probability threshold: If the threshold is increased to guarantee more reliable predictions, the risk of discarding

correct predictions with a low probability rises. Figure 5.6 c) shows the F-score and reveals the characteristics of this trade-off: The F-score rises as the threshold increases from 0 to 0.4. Up to this threshold, the gain from a higher precision outweighs the loss in recall. However, for a threshold higher than 0.4, the loss in recall becomes more severe, so that the score is negatively affected. The best trade-off performance can be achieved when choosing $p = 0.4$ as a probability threshold. In this case, a F-score of 0.86 can be achieved which guarantees the best trade-off decision.

The evaluation results for the *Eventually* operator ($F^{\leq t}\varphi$)) are plotted in Figures 5.7 a) - c). We can see that the precision significantly gains already for lower thresholds $p > 0$ as demonstrated in Figure 5.7 a). This is due to the special prediction semantics of the F operator, which predicts location visits at any state reachable within the given time constraint over multiple possible state transitions. Thus, a location visit may be found also in further states (not only in the next states), so that the chance to satisfy a prediction overall increases. As a result, there is a positive effect on the precision when granting more freedom about the states where predictions may become true. At the same time, the recall is strongly reduced by an increasing threshold as shown in Figure 5.7 b). Since future location visits at states further away are incorporated, a significant amount of predictions has only a minor probability. The reason is that the occurrence probability usually degrades with each additional transition taken, as the transition probability is distributed among several transition at a state. This causes a lot of FNs in total, so that a significant amount of correct predictions is discarded. This observation is also reflected in Figure 5.7 c), which shows the F-score for different probability thresholds. For thresholds $p > 0.2$ the high loss in recall dominates the gain in precision. Due to this, the Eventually operator is more sensitive to the choice of the probability threshold compared to the Next operator. The best configuration is found for $p = 0.2$. For this threshold, a F-score of 0.87 can be achieved which exhibits the best trade-off characteristic over all configurations.

5.7. Related Work

Over the past years, there has been active research to improve the capabilities and performance of context prediction systems [BZ10]. However, while context prediction accuracy has received a lot of attention, the expressiveness of context prediction has not been in the focus of this research. In the following, we give an overview of existing prediction systems in terms of the type of predictions that they can compute to point to their limited power of expressiveness. Then, we discuss methods from the area of model checking as a powerful tool to fill this gap, which has not been considered so far in prior context prediction research.

The context prediction system of Maryhofer [May04] includes layers of context recognition and prediction in order to learn about current and future context states of a

user. For this purpose, context prediction is defined as a problem of extrapolating a time series into the future, i.e. given a series of previous context states the next context occurrence shall be predicted. In contrast, PreCon allows for queries which target context changes beyond the next state. Further, since PreCon's predictions may involve powerful logic-based expressions, our approach is superior in terms of query expressiveness.

Similarly, further prediction schemes have been proposed to infer a user's next context state. Common to Markov models [KM09, SKJH04, DH98], compression algorithms [SKJH04, GC07], or n-gram tries [HS07] is a statistical approach to learn patterns from a series of past context occurrences that reveal typical context transitions. All these approaches represent predictors which can be used to anticipate the next context state of a user. For instance, fixed order Markov models rely on a limited number of preceding state transitions. In contrast, compression algorithms derive the prediction from a varying number of preceding transitions. Nonetheless, these approaches can only be used to predict the next most probable context and queries of higher expressiveness are not foreseen.

Machine learning is traditionally used to address classification problems. In essence, classification can be applied as a generic method when the problem is to find a fitting category for a given observation. Anagnostopoulos et al. have shown how to adopt these methods to predict the future location of users, training a classification model from sequences of past location visits [AAH+09]. The classification task is then to assign a next location visit based on the last user movements. For this purpose, they have evaluated different classifiers such as the Bayesian classifier or Decision Trees in terms of the resulting prediction accuracies. However, only the most probable next context in a single-dimensional context space of locations can be predicted. In particular, queries for temporal relations in a multi-dimensional context space are not supported.

Lee and Hou study the mobility characteristics of users across the Dartmouth campus network [LH06]. In order to describe the users' typical movements pattern, a Semi-Markov Model is used to represent temporal aspects in the mobility behaviour, i.e. the typical durations of associations of user devices with wireless access points deployed at the campus. Moreover, an approach is developed to estimate the access point to which a user is connected at a future time, for which the temporal association patterns are used as a basis for the prediction. However, beyond the specific prediction in the focus of their work, a sophisticated query language that allows for formulating predictions with arbitrary temporal relations and logic-based expression is not considered. PreCon is the first system to investigate the application of temporal logics as a powerful and expressive query language for context predictions.

The algorithms for the prediction of context information used in PreCon originate from the field of stochastic model checking [KNP07]. Stochastic model checking algorithms for Continuous Time Markov Chains have been initially studied by Baier et al. [BKH99]. In later years, Lopez et al. have proposed extensions of these algorithms for Semi-

Markov Models [LHK01]. While we leverage on these algorithms as a basic statistical framework for the design of PreCon, prior model checking approaches have been devised to support the off-line analysis of static software systems. For the purpose of context prediction, algorithmic extensions are required to consider run-time semantics and answer predictions related to the current state in human behaviour. To this end, we have developed an approach to condition the predictions on the user's running state dwell times. This allows for more accurate real-time predictions since the dwell time significantly influences the probability of a state change as time elapses. While model checking systems are usually applied to study critical system properties (e.g. the safety guarantees of traffic light circuits), we have used the metrics precision and recall from the area of information retrieval to gain insight into the predictability of human behaviour. This shows that, by applying model checking for context prediction tasks, novel research questions arise which have not been addressed by any prior model checking system.

5.8. Summary

In this chapter, we have presented PreCon, an novel context prediction approach that combines stochastic model checking techniques with statistical learning to allow for more powerful and expressive forecasts. The key components of PreCon are a time-dependent stochastic user model, an expressive temporal-logical query interface and novel context prediction algorithms. More precisely, PreCon uses a Semi-Markov Model (SMM) to incorporate an explicit model of the temporal aspects in a user's behaviour. Further, PreCon is based on temporal logics as a query language which facilitates predictions with logical expressions and rich temporal semantics (e.g. will the user be at location x within the next 10 minutes). For prediction inference, we have extended well-known model-checking techniques to deal with the special requirements of predicting human context where forecasts need to be conditioned on information about the user's current state and dwell time. We conclude that the combination of SMMs, a query language based on temporal logic, and the online learning approach significantly increase the expressiveness of context prediction system and therefore represents a significant contribution to the state-of-the-art in this area.

We have evaluated PreCon based on a real-world case study from the area of health-care using metrics from information retrieval. In our evaluation, we have shown that expressive temporal predictions can be made with high precision-recall characteristics, enabling applications to access future context information with effective trade-off decisions.

Part III.

Context Prediction in Mobile Systems

Chapter 6

Mobile Sensing Applications

In this chapter, we propose and study tailored context prediction approaches to improve the performance of context-aware mobile systems. We focus on a new class of mobile systems known as mobile sensing applications, which have recently emerged as a result from the versatile sensing capabilities of powerful human-carried devices [CEL+08, LML+10]. In mobile sensing applications, streams of context updates are captured on the users' mobile devices and shared with remote parties to enable new forms of collective context-aware services. This has led to a paradigm change, where mobile users are no longer mere consumers, but represent active producers of context data. However, mobile sensing applications are facing severe resource constraints of mobile devices, which makes their development a significant challenge. As a result, the provision of suitable methods to improve upon the resource usage of these applications, especially in terms of the underlying energy consumption, has moved in the focus of current research [RMM+11, BDR12b].

In the following, we will delve into the specifics of mobile sensing applications to lay the foundation for novel context prediction methods approaches which are tightly integrated with the operation of these applications. First, we start by discussing mobile social networking as a possible application scenario which is attractive to a large user basis [MLF+08]. This discussion will reveal that a significant overhead of energy consumption is caused by these applications in real environments which is critical for practical deployments. Then, we present a generic model of mobile sensing applications which allows us to abstract their general behaviour in terms of basic operations for sensing and distributing context data. Subsequently, we analyse the energy characteristics of mobile data communication to motivate the need for more sophisticated protocols tailored to the requirements of mobile sensing applications.

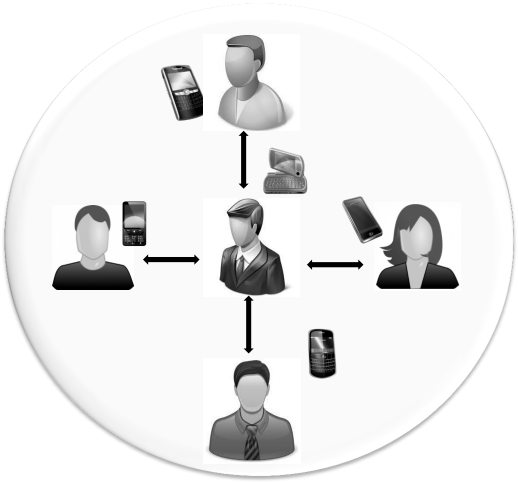


Figure 6.1.: Online Mobile Social Network comprising a mobile context producer (middle) and four context consumers in his social circle

6.1. Application Scenario

The area of mobile sensing has spawned a range of innovative applications that allow mobile users to contribute sensor data from their mobile phones [CEL+08, LML+10, RH10]. One such application relates to online social networking, where platforms such as Facebook or Google+ have attracted millions of users over the past years, allowing people to connect with their friends in order to improve their social experiences. In the future, it is envisioned that online social networks can become even more seamlessly integrated into everyday life by effectively exploiting mobile sensing technology to design new ways of engaging in social communication (cf. Figure 6.1). In particular, while current social network systems rely on manual updates of the users, mobile sensor-enabled devices allow for extracting the users' context in an automatic manner to share context updates with their friends in real-time. This will further improve the flow of information among users and allow friends to receive constant updates about the activities of their social links as soon as they occur (where they are and what they are doing). As the updates are triggered automatically by changes in the users' context, social network users are thus able to share their activities without spending time and laborious effort to enter this information explicitly.

Driven by this vision of future online social networks, researchers have developed first prototypes to detect the users' presence information and push this information to a user's social friends [MLF+08]. Based on experiments with a real-world deployment of the application prototype, the performance of the application could be monitored and the experience of users could be surveyed [MLF+08]. While the users' feedback confirmed the high value of the application to provide novel social experiences, significant technical

limitations have been encountered that hamper a wider adoption of this technology. Specifically, it has been reported that the battery of standard mobile devices is already fully drained within six hours, caused by the frequent updates of the users' context over mobile communication networks [MLF+08]. Hence, there is a significant gap between the potential value of mobile sensing applications and limitations in terms of the underlying energy overhead. In the remainder of this thesis, we look at this problem in detail and propose tailored context prediction methods to allow these applications to be executed in an energy-efficient manner, while preserving the usability of the application as much as possible.

While the idea of mobile social networking perfectly illustrates the technical challenges associated with mobile sensing applications due to the high interest of a large user basis and the always-on mode of operation, other mobile systems (e.g. mobile health guides) are similar in nature and also suffer from critical constraints in terms of limited energy resources. In order to account for this, we develop in the next section a more general model of mobile sensing applications that allow us to apply our concepts and algorithms to a wide spectrum of different scenarios.

6.2. System Model

Nowadays, humans carry powerful mobile devices such as smartphones which provide rich opportunities for context-aware applications. Such devices are equipped with a set of versatile on-board sensors (e.g. GPS, camera, acceleration, compass, etc.) for detecting context data in real-time. The sensors can be used to continuously capture time series of low-level sensor data such as sound, acceleration or Wi-Fi signals. Based on the physical sensor readings, information about the high-level context of the user can be inferred (e.g. activity or symbolic location). For instance, it has been shown that the user's mode of locomotion (e.g. sitting or standing) can be recognized from acceleration samples using statistical frameworks such as the Naive Bayesian classifier [BI04].

By classifying raw sensor data into context states, the sensing process yields high-level context of discrete nature, representing the different states in which a user can be. Context states are associated with labels (e.g. "sitting" or "walking") and thus provide intuitive information that is easy to interpret for applications and users. Formally, a context state is denoted by $s_i \in S$, where S is the state space. Context states can be multi-dimensional, in which case they represent tuples of different context types. For instance, $s_i = (\textit{"office"}, \textit{"sitting"}) \in S$ describes a state, where the user is sitting (classifying his activity) in his office (classifying his location).

For detecting the current context state of a user, the sensors are sampled with sensing interval Δt_s. Due to this time-discrete sensing scheme, each context s_i is assumed to be valid for the interval $[t, t + \Delta t_s)$, starting from its time of sensing t and ending at the next sensing cycle $t + \Delta t_s$. Continuous sensing results in a sequence of user context

states $\{s_{t_1}, s_{t_2}, \ldots, s_{t_n}\}$, where s_{t_1} denotes the first and s_{t_n} the last sensed context and s_{t_i} refers to the context available at time $t_i = i \cdot \Delta t_s$ for $1 \leq i \leq n$. Please note that two consecutive context states s_{t_i} and $s_{t_{i+1}}$ may be equal ($s_{t_i} = s_{t_{i+1}}$), if the user context has not changed in the meantime.

Additionally, the devices are equipped with a cellular wireless communication interface (e.g. UMTS or GPRS) for mobile Internet access. The communication interface is used to disseminate information about the changing context of mobile users. Mobile devices can take different roles in the dissemination process. We distinguish between a producer, whose context is locally monitored on the device, and remote consumers, which are interested in context changes of the producer. Formally, let the producer context at any time $t \in T = \{t_1, \ldots, t_n\}$ be denoted as $p(t) \in S$, while the producer's context known to the consumer is referred to as $c(t) \in S$. In order to synchronize a consumer with a producer (so that $p(t) = c(t)$ is guaranteed), information about the producer's local context state must be forwarded to the consumer. For this purpose, update protocols are employed which decide about when the next update message should be triggered. We will study various update protocols to accomplish this task in the next chapter.

As mobile devices are equipped with batteries that have only limited capacities, energy consumption is a critical factor for the usability of the devices. The lifetimes of the devices' batteries therefore strongly depends on the overhead of energy-intensive operations. Unfortunately, mobile sensing applications require costly sensing and communication operations. Especially, frequent updates of context information over cellular radio networks may substantially drain a device's battery [MPF+10]. To elaborate on this aspect more in detail, the cellular power management characteristics of mobile devices are analysed in the next section.

6.3. Energy Characteristics of Mobile Data Communication

The cellular power management of mobile devices reveals important implications for the design of mobile sensing applications. In the following, we analyse the relation between the inherent requirements of mobile sensing applications and the energy overhead incurred by mobile communication. This analysis reveals a problematic mismatch with a negative impact on the energy overhead: while mobile sensing applications are characterized by frequent small transmissions to distribute observed context changes, cellular network interfaces are based on a power management strategy which favours sporadic transfers of large chunks of data.

More precisely, the total energy spent on mobile devices for transmissions over cellular networks originates from different sources of energy consumption [BBV09]. In order to facilitate the physical data transfer, transmission energy is required which depends on the amount of data exchanged over the wireless channel. Consequently, transfers of small chunks of data do not consume a significant amount of transmission energy.

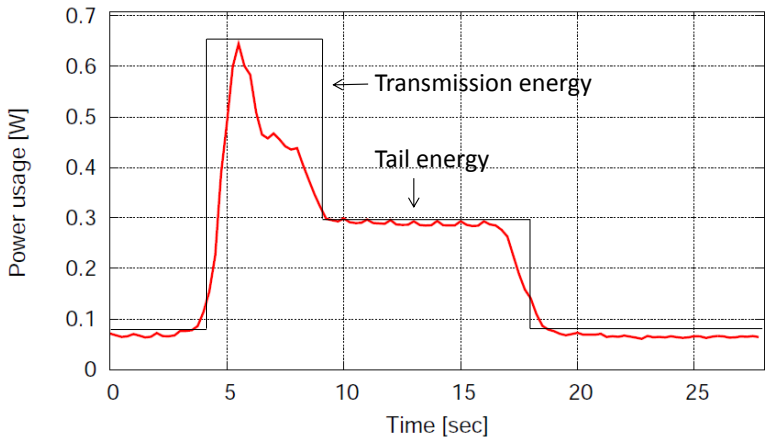


Figure 6.2.: Energy consumption characteristics for mobile data communication over cellular networks [MPF+10]

However, once the data transfer has been completed, the interface remains in a high power state for an additional period of time which is referred to as tail time. This is done to keep the interface powered up in expectation of additional transmissions which may occur subsequently. For this reason, as part of the total energy consumed for a data transfer, the so-called tail energy is included that is spent in the tail time. As an example, Figure 6.2 plots the energy profile for a data transfer of 100 bytes on a Nokia N95 smartphone over a GPRS network. As the sampled power usage shows, there is a significant period of time during which the wireless interface remains active after the completion of the data transmission during which additional energy is consumed.

As the above discussion implies, the design of current cellular network technologies is optimized for high-volume data transfers. For transferring high volumes of data, the tail energy is insignificant compared to the energy consumed by the actual data transfer. However, mobile sensing applications require frequent updates of small data items, which invalidates this assumption. For instance, in case of GPRS the tail time is 6s during which 0.025J/s is spent [BBV09], resulting in a tail energy of 1.5J. Assuming a small message size of 1KB which is enough to store the context update information (i.e. a context state), the total energy amounts to 3.2J that is consumed per message. In such a scenario, the tail energy takes up a significant proportion of the required energy (47%), so that transferring small-sized messages causes considerable energy overhead. For instance, sending 200 update messages per hour would result in 640 J/h for transferring only 200KB of data*. As this discussion shows, mobile sensing applications are highly susceptible to a high energy consumption due to sending frequent

*For this calculation, we assume that every message causes the complete tail energy to be consumed, which occurs each time the next context state is not sent until the complete tail time has elapsed. This assumption is met by current context sensing approaches and their underlying sampling frequencies [MPF+10]

update messages which challenges the practical usability of mobile phones. Therefore, new approaches are required for rendering this update process more efficiently. In the following chapter, we will explore how context prediction methods can help to design update protocols for mobile sensing application with improved energy consumption characteristics.

Chapter 7

Energy-efficient Context Update Protocols using Context Prediction

7.1. Introduction

In this chapter, we present an approach for the integration of context prediction methods into update protocols for distributing discrete context data. Update protocols have a long tradition in mobile computing to control the quality of context data and message overhead of distributed applications [BD99, LR01]. Given the limited resource constraints of mobile devices, the design of efficient update protocols is an important research question. In prior work, prediction-based update protocols have been devised which can improve the protocol efficiency. For instance, location-based applications can employ dead-reckoning techniques to track the geographic positions of mobile objects under given spatial deviation bounds [WSCY99]. With dead-reckoning, a spatial movement prediction function is used to estimate the future trajectory of a mobile user for reducing the number of updates caused by changes in the user's position. While update protocols for tracking user positions are well-understood, an in-depth investigation of prediction-based methods for the distribution of discrete context data is lacking in current research.

Compared to geographic position information, discrete context can be associated with arbitrary labels to describe the semantics of a user's behaviour, e.g. for the classification of a user activities [MLF+08]. As a consequence, discrete context states can encode arbitrary meanings, for which assumption about user mobility in geographic space do no hold as exploited by existing location-based update protocols [LR01]. Therefore, new models and techniques are required to implement prediction-based methods for update protocols aiming at discrete context. In particular, the properties of the discrete context model need to be closely reflected in the update protocol. First, a prediction model is needed which can represent the evolution of a user's behaviour as a description

of context state dynamics. Second, an accuracy metric is required which is able to express the correctness of state-based context data. Third, relevant decision criteria and deviation thresholds need to be established, where the state of predicted and actual context occurrences are compared for triggering update messages.

As a response to this challenge, we propose in this chapter protocols for delivering energy-efficient updates of discrete mobile user context using context predictions. The protocols are designed for allowing applications to trade-off the energy consumption on a mobile device against the context accuracy observed by remote consumers. While the energy consumption results from the overhead of message updates sent by a producer, context accuracy is based on a boolean metrics which measures the fraction of time over which context states at a producer and consumer match. In order to improve the trade-off characteristics, we propose an approach for maintaining a high level of context accuracy without the overhead for a large number of update messages. To this end, we propose four different context predictors that can be embedded into update protocols to forecast evolutions of context changes. Using the predictions, a producer's current context can be estimated at a consumer from an outdated context update which was received in a prior update message. We present different variants of Markov predictors which apply specific strategies for computing a prediction, exploiting the state transition probabilities and dwell time characteristics encoded in the Markov model. To integrate the context predictions into the update process, two basic update protocols are devised, a time-based and a deviation-based protocol, which allow for controlling the update triggers to determine when the next update has to be sent.

In our evaluation, we analyse the inherent trade-off between energy consumption and context accuracy of our protocol variants for a real-world user trace. The results gained from our evaluations are twofold: in scenarios where the real-world behaviour matches the statistical assumptions made by Markov statistics, the trade-off can be significantly improved by exploiting successful predictions of the future context changes. In contrast, in case the real-world behaviour does not obey these assumptions, the protocols are susceptible to predictions inaccuracies. By further analysing this problem in detail, we gain valuable insights for the design of a more robust prediction-based update protocol that is presented in the next chapter as a follow-up approach.

The rest of the chapter is organized as follows. In Section 7.2, we give a formal description of the problem targeted by our approach. Then, we give a basic overview of the prediction-based update process in Section 7.3. In Section 7.4, we propose two different protocol variants to control the transmission of context updates. Based on our prediction model which we introduce in Section 7.5, we then present in Section 7.6 four novel context predictors. Thereafter, we present our evaluation results in Section 7.7 and discuss previous update protocols from related work in Section 7.8. Finally, we conclude with a summary in Section 7.9.

7.2. Problem Statement

In this chapter, we study a fundamental problem in the design of mobile sensing applications. The goal is to devise energy-efficient update protocols for transferring steams of context data from mobile devices to interested consumers. For the design of such protocols, we focus on the trade-off between the energy costs for sending context updates and the context accuracy which can be observed by interested consumers. In the following, we further elaborate on this problem in more detail.

As discussed in Section 6.2, mobile devices can be either producers or consumers of context data. At a producer, the current context state is provided at periodic sampling intervals Δt_s. For instance, recognizing an activity requires collecting sufficient evidence about variations of the sensor signals over an interval $\Delta t_s = 4s$ [MPF+10]. Consequently, we assume that the context state at a producer evolves as a time-discrete process with steps $T = \{i \cdot \Delta t_s | 1 \leq i \leq n\}$. Formally, the context state of producer at any time $t \in T$ is defined as $p(t)$.

In contrast, the consumer resides at a remote site and has no direct access to the producer's context updates. Therefore, a consumer needs to be informed about relevant context changes via wireless communication. The context of which the consumer is aware of at any time $t \in T$ is denoted as $c(t)$. In order to influence $c(t)$, update protocols are applied that elicit update messages to report about context changes. Applying a specific update protocol has important implications in terms of the energy consumption at both parties and the context accuracy at the consumer.

Context accuracy rates the exactness of the context data seen by a consumer. Depending on the underlying update protocol, certain context updates may be suppressed so that the context $c(t)$ experienced by a consumer may differ from the real context $p(t)$ at a producer. Formally, this can be measured as

$$\lambda(t) = \begin{cases} 1, & \text{if } p(t) = c(t) \\ 0, & \text{else} \end{cases} \tag{7.1}$$

, where $\lambda(t)$ yields whether the producer and consumer states match at time t. While $\lambda(t)$ is a measure of the instantaneous context accuracy at time t, we can use it for deriving a complete view on the context accuracy. Normalizing (7.1) over the application lifetime of $|T|$ samples of context states, results in the average accuracy

$$ac = \frac{\sum_{t=1}^{|T|} \lambda(t)}{|T|} \tag{7.2}$$

as a basic measure for the consumer-side context accuracy. Thus, ac indicates the fraction of time for which the consumer is in the correct state as a value in $[0, 1]$. In this regard, it gives insight about the extent to which the consumer experiences a loss of information.

At the same time, achieving a high context accuracy requires frequent context updates via wireless cellular radio channels (e.g. GPRS). Each time an update is sent, we pay e_u amount of energy for its transmission. To study the overhead of wireless data communication, let the set of all updates made be denoted as $U \subseteq T$. The energy consumed for wireless communication at any time t is thus dependent on the update decisions taken and can be expressed as:

$$\mu(t) = \begin{cases} e_u, & \text{if } t \in U \\ 0, & \text{else} \end{cases} \tag{7.3}$$

Based on the instantaneous energy consumed at a certain time $t \in T$, this leads to an energy consumption rate of

$$ec = \frac{\sum_{t=1}^{|T|} \mu(t)}{|T| \cdot \Delta t_s}, \tag{7.4}$$

which reflects the energy consumed per time (J/h) over the complete application lifetime.

There exists an inherent trade-off between the required energy consumption *ec* and the resulting context accuracy *ac*. On the one hand, with energy being a strictly limited resource on mobile devices, a high rate of context updates may over-stress the devices' batteries. On the other hand, not sending an update message impedes the context accuracy at a consumer who expects accurate knowledge of a producer's context. Therefore, the goal of our work is to design energy-efficient update protocol which are able to improve this trade-off in an effective manner. Specifically, these protocols should achieve a reduction in energy consumption *ec*, while mitigating the negative impact on context accuracy *ac* at the same time. In the following, we propose and evaluate different strategies to achieve this goal.

7.3. Approach Overview

We propose different prediction-based protocols to allow for energy-efficient context updates. A taxonomy of our protocols is shown in Figure 7.1. Fundamental to our approach are two basic update protocols, a time-based and a deviation-based protocol, that can be used to control the overhead of sending update messages. With these protocols, the set of context updates which are delivered to consumers can be selectively chosen. By a suitable configuration of these protocols and their update decisions, it is feasible to effectively influence the energy/accuracy trade-off which is critical to mobile sensing applications.

Supplementary to the basic update protocols, we have developed four different predictors which can be integrated into these protocols. Using these predictors, both producer and consumer independently determine the producer's most recent context change in periods of Δt_s. The context predictors are designed to use the latest information from

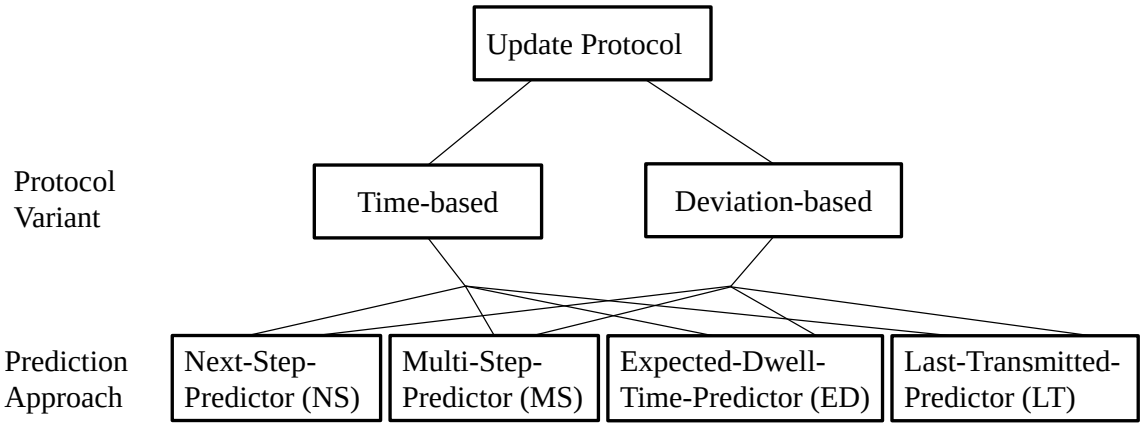


Figure 7.1.: Overview of energy-efficient update protocols

the last update message to estimate a producer's current context state. However, as this last update message may have been sent some time ago, the producer's context state needs to be predicted over several intervals.

Consumers exploit these predictions to approximate the current context of a producer. This way, producers do not have to send every context change, since consumers are empowered to infer these updates on their own. In contrast, producers use the predictions to monitor the update process and verify what context state is currently suspected by a consumer. If a deviation is recognized which cannot be tolerated, the producer triggers a transmission to update the consumer with the most recent context change that was observed. The strictness of how much synchronization is necessary and the extent to which deviations are tolerated is controlled by the underlying update protocol.

Hence, accurate predictions amplify the efficiency of a prediction-based update protocol: On the one hand, the context accuracy is preserved at a consumer due to the ability of knowing about the producer's context in spite of missing updates. On the other hand, whenever the predictions match, a lower number of context updates is transmitted by the producer, so that less energy is consumed for distributing the updates over the wireless communication channel.

In order to anticipate context changes, a prediction model is required which describes typical patterns inherent to the producer's context. This model is learned by the producer and shared with the consumer, initially when starting the application for the first time and in case of radical changes in the user's behaviour (e.g. when leaving university and starting a job which may introduce new context states). Note that in contrast to prediction-based update protocols for geographic user positions [WSCY99],

there is no need to update the prediction model with every context update. In case of a discrete context model, the dynamics of user behaviour can be encoded with states and state transitions. The prediction model therefore does not only enable forecasts related to the last known user context, but is also valid for inferring new predictions whenever the context is updated. Since in current research no longitudinal data is available to explore long-term changes in user behaviour, we focus in this work on the required techniques to make prediction-based update protocol for discrete context generally possible. The proposed techniques are valid in any scenario where prediction-based updates shall be applied irrespective of the strategies chosen to refresh the prediction model.

7.4. Basic Update Protocols

In this section, we present two different update protocols, a time-based and deviation-based protocol, for controlling the update process on a mobile device. Both update protocols are designed in a way to allow for the integration of context predictions to reduce the number of context updates. As the protocols are agnostic to the prediction inference algorithm, they can be flexibly combined with any predictors we introduce in Section 7.6. Instances of the protocols run on both the producer and consumer. However, since the update decision is enforced by a producer, we focus on this part subsequently since it involves more decision-making logic. The consumer part essentially consists of the prediction task, which is discussed in detail later.

7.4.1. Time-based Update Protocol

The time-based update protocol follows a simple idea to control the update message overhead. The protocol is executed regularly in constant update periods Δt_p. Each time the update period elapses, the current context of the producer is reported to the consumer. In between two consecutive updates, the producer's context may evolve while no update message is sent to the consumer. In order to compensate for any suppressed update, the consumer attempts to predict the correct context state of a producer. A formal description of the protocol is given in Algorithm 5. The protocol starts with an initial context update (lines 1-2). Then, the producer waits for the begin of the next update cycle. In each cycle, the current producer context is compared to the predicted consumer context (lines 4- 6). An update message is sent only if the consumer gains new information. Hence, if the producer's context is different from the prediction, a context update is triggered that is delivered to the consumer (lines 7-9).

The benefit of the update protocol is the simple control of the expected overhead of context updates. Using an update period of Δt_p, the frequency of updates is limited to at most $\frac{1}{\Delta t_p}$ in the worst case. As redundant context updates are suppressed, the number

Algorithm 5 Time-based Update Protocol

```
1: s_P ← sense producer context
2: send update message(s_P)
3: while true do
4:     sleep(Δt_p)
5:     s_P ← sense producer context
6:     s_C ← predict consumer context
7:     if s_C ≠ s_P then
8:         send update message(s_C)
9:     end if
10: end while
```

of context updates may be even lower. Consequently, the elapsed period between two updates may be any multiple of Δt_p ($k \cdot \Delta t_p$ with $k \in \mathbb{N}$). At the same time, the protocol limits the duration that the consumer spends in the wrong context state. Due to the constant update period, the consumer knows at least every Δt_p about the real context, i.e., $\forall t \in \{t_1 + (i \cdot \Delta t_p) | i \in \mathbb{N}_0\} : p(t) = c(t)$ holds. However, at the same time, context changes between the context updates may be missed if they are not correctly predicted by the consumer.

Next, we present another update protocol that acts on a series of context occurrences instead on single time instants. For this protocol variant, the divergence of the consumer context from the real producer context is employed as underlying update criterion.

7.4.2. Deviation-based Update Protocol

This protocol is based on a tolerance towards the duration for which a consumer may miss the real context. The consumer's tolerance is specified as a time bound $\Delta t_d \in \{i \cdot \Delta s | i \geq 0\}$, which indicates the maximum time span for which the current context assumed by the consumer may diverge from the real context at the producer. Formally, this bound guarantees that $\forall t \in [t_1, t_n] : (p(t) \neq c(t) \ \rightarrow \exists \delta \leq \Delta t_d : p(t + \delta) = c(t + \delta))$, i.e., the period of inaccuracy is strictly limited by duration Δt_d.

The protocol that satisfies this criterion is described in Algorithm 6. Initially, a context update is triggered when the application starts (line 1-2). Then, in each sensing interval, the producer obtains the currently valid context (line 6) and makes a prediction about this context (line 7) to share the same view as the consumer. The duration for which the producer and consumer context diverges is recorded in t_{error}. Each time the producer context (s_P) is different from the context predicted by the consumer (s_C), t_{error} is increased to reflect the grown time of divergence (lines 7-8). t_{error} is reset to zero, as soon as the two context states become equal ($s_P = s_C$) (line 11). Whenever t_{error} would exceed the time bound Δt_d, a context update is sent. Afterwards, the producer and

Algorithm 6 Deviation-based Update Protocol

```
1: s_P ← sense producer context
2: send update message(s_P)
3: t_error ← 0
4: while true do
5:     sleep(Δt_s)
6:     s_P ← sense producer context
7:     s_C ← predict consumer context
8:     if s_P ≠ s_C then
9:         t_error ← t_error + Δt_s
10:    else
11:        t_error ← 0
12:    end if
13:    if t_error > Δt_d then
14:        t_error ← 0
15:        send update message(s_P)
16:    end if
17: end while
```

consumer states are synchronized again so that t_{error} can be set to zero (lines 13-16). Note that if $s_P \neq s_C$ holds for less than Δt_d time units, no update will be sent. This means that the context may change for a short period of time and then return to its previous state without the consumer being notified about this. In the most restrictive case of $t_d = 0$, no deviation is tolerated. In this case, t_{error} would exceed t_d once a single deviating context state is sampled which could not be successfully predicted. Therefore, $t_{error} > t_d$ is applied as an update criterion to enforce that an update is sent if the decision to suppress an update would violate the deviation bound.

Similar to the time-based protocol, the protocol provides an efficient way to control the message overhead. Due to the bound on the maximum duration of inconsistency, the update frequency does not exceed a rate of $\frac{1}{\Delta t_d}$. However, this represents a worst case estimation which assumes that the consumer continuously makes wrong predictions. If a more effective prediction accuracy is achieved, the update rate will be significantly lower as a consequence from the correct alignment of the producer and the consumer. In every case, the protocol guarantees that $\forall t \in [t_1, t_n], \exists t_i \in [t, t + \Delta t_d] : p(t_i) = c(t_i)$ holds, i.e., that the consumer is aware of the correct context state at least every Δt_d time units. In contrast to the time-based protocol, where the updates are triggered in periodic intervals, the deviation-based protocols reasons over a series of context occurrences for measuring the alignment of producer and consumer. As a consequence, the time interval between two updates is variable. The particular effects on the energy/accuracy trade-off of this protocol and the difference of the time-based and the deviation-based protocols become visible as part of our evaluation which is discussed in Section 7.7.

7.5. Stochastic User Model

The context predictors which are used to enhance the update protocols in this work are based on a common model of the user's behaviour. Before going into detail about the different prediction algorithms, we introduce this model first. For the representation of the user behaviour dynamics, we rely on a discrete Markov model [How71a]. According to this model, the user is in a discrete state at each point in time. The evolution of user states is considered as a stochastic process, where the state of a user at time t_n is captured by a random variable X_{t_n}. The Markov model encodes the user behaviour as a transition matrix M with entries $p_{ij} \in [0, 1]$. The entry $p_{ij} = p(X_{t_{n+1}} = s_j | X_{t_n} = s_i)$ denotes the probability for a transition from state s_i to s_j and $\forall s_i \in S : \sum_{s_j \in S} p_{ij} = 1$. Based on M, not only the probability for a single transition, but also for several transition over multiple time steps can be derived. Specifically, $p_{ij}^k = p(X_{t_{n+k}} = s_j | X_{t_n} = s_i)$ defines the probability for reaching the state s_j from state s_i in exactly k steps for $k > 0$. This probability can be computed as the k-th power of M. For this purpose, we inductively define $p^k = p^{k-1} \cdot p$ and $p^1 = p$. The entries p_{ij}^k thus denote the transition probabilities for a k-step transition from s_i to s_j.

We explicitly allow self-transitions in M, i.e., $\forall s_i \in S : p_{ii} \geq 0$. Thus, the user may either be in the same or in a different state after a transition is taken. Self-transitions are required to express the fact that a user may dwell in his state, e.g. when executing an activity such as sitting. Moreover, let the stationary probability distribution of the Markov model be denoted as $\pi : S \to [0, 1]$ [How71a]. Then, $\pi(s_i)$ (also denoted as π_i) encodes the stationary probability of state s_i, i.e., the probability with which s_i occurs in an infinite time series of context occurrences. The stationary distribution gives valuable insight into the long-term behaviour of users, i.e., the extent to which particular states are occupied if the transition system is followed infinitely. Further, the Markov model reveals the dwell time distribution $dw(s_i)$, which expresses the probability of the time that a user spends in a particular state s_i. The dwell time distribution can be derived from the self-transition probabilities associated with a state. The probability to stay for a number of $n \geq 1$ times in the same state s_i is indicated by the geometric distribution $P(dw(s_i) = n) = (1 - p_{ii}) \cdot p_{ii}^{n-1}$, and its expected value is given by $E[dw(s_i)] = \frac{1}{1-p_{ii}}$ [How71a].

We consider a classical discrete Markov model as an effective means for predicting future user behaviour. This model is based on the assumption that the probability for entering a future state only depends on the current state. Such a Markov model can be learned from traces of past user behaviour, containing records of sequences of context occurrences. Due to the periodic sensing scheme, a context is recoded every Δt_s times in a context trace. The transition probabilities of the Markov model can be derived by counting the frequencies of pairs of context states (i.e. of a state and its successor) in the trace. p_{ij} then represents the probability for a transition from s_i to s_j to occur in the next sampling period Δt_s. Analogously, the k-step transition probability p_{ij}^k refers

to a transition from s_i to s_j to happen after k sampling periods (i.e. a time duration of $k \cdot \Delta t_s$). As we have shown in Section 4.2.1, also higher-order Markov models, which consider a longer history of previously visited states (beyond the current state), can be represented with the same approach. For this purpose, a sequence of n context elements can be abstracted as a single context state in the model. Thus, our stochastic user model is able to encode a range of statistical properties underlying different Markov models that we can exploit for prediction.

7.6. Context Predictors

In this section, we present four different context predictors that can be integrated into our update protocols to improve the energy/accuracy trade-off. All these predictors are devised to address the same prediction task, i.e., forecasting the producer's current context using a Markov model as an underlying prediction model. However, each predictor has a unique way of exploiting the specific properties of the Markov model to make a prediction. Hence, the predictors are basically competitors, and in our evaluation (see Section 7.7) we quantitatively compare the predictors' performance to support energy-efficient updates.

As described in Section 7.4, our update protocols rely on a prediction of the producer's current context state to lower the update rate. The prediction task can be formally stated as follows. The goal of the prediction is to forecast the current context state $f(t_n)$ valid at time t_n based on the most recent context state $s_{t_{n-k}}$ that has been supplied to the consumer at time t_{n-k}, where $k > 0$. Note that depending on the value of k there may be a significant time gap in between t_n and t_{n-k}, where no additional update has been sent to the consumer. Consequently, the prediction has to estimate successive state transitions over several sampling intervals to fill the gap in knowledge about what changes happened to the producer's context. We also call k the prediction horizon to stress the number of successive forecasts which have to be made. In the following, we present four different predictors that take a specific approach to compute such a prediction.

7.6.1. Next-Step-Predictor (NS)

The NS predictor is a simple predictor which exploits the information stored in a Markov model to implement a greedy search strategy. The idea is to avoid complex computations by identifying a single path through context states over several transitions which is followed with a high probability. Initially, the prediction of the next missing context state can take advantage of the knowledge of the last transmitted context $s_{t_{n-k}}$. For this purpose, let $idx(s_{t_{n-k}})$ be a function that returns the index of a given state $s_{t_{n-k}}$ as it is stored in the Markov model. The prediction for the next transition is then defined as

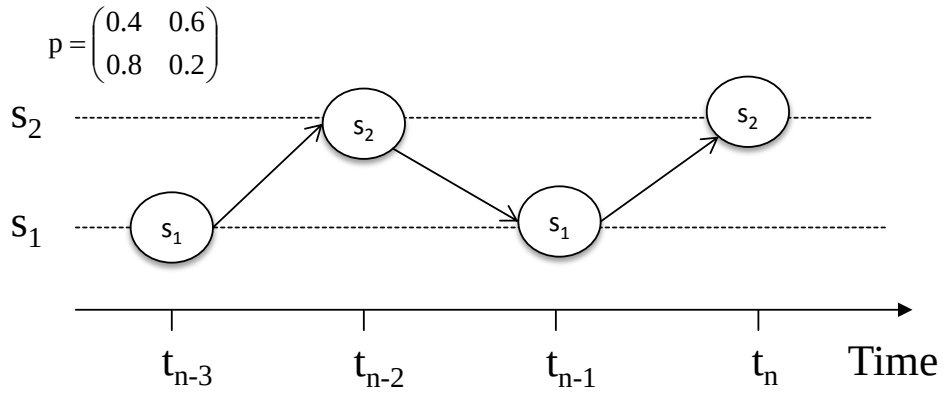


Figure 7.2.: Example of the Next-Step-Predictor (NS)

$$f(t_{n-k+1}) = s_r \text{ with } r = \arg \max_{j \in \{1,\ldots,n\}} \{p_{idx(s_{t_{n-k}}),j}\}$$

i.e., the prediction $f(t_{n-k+1})$ at time t_{n-k+1} is given by the one which can be reached over the transition with the highest probability from the last known state $s_{t_{n-k}}$. While for $k = 1$ the algorithm terminates here, for cases of $k > 1$ predictions over several sample intervals need to be computed. For this purpose, the NS predictor relies on the last prediction to extrapolate the user's behaviour further into the future. The prediction at time t_{n+i-1} for $2 < i \leq k$ is then defined as

$$f(t_{n-k+i}) = s_r \text{ with } r = \arg \max_{j \in \{1,\ldots,n\}} \{p_{idx(f(t_{n-k+i-1})),j}\}$$

i.e., the prediction for $f(t_{n-k+i})$ is given by the state which is reachable over a transition with the highest probability from the last prediction $f(t_{n+k+i-1})$. Since such a prediction depends on a prior prediction in turn, the uncertainty further increases with missing context updates and longer prediction horizons.

The principle of this prediction algorithm is illustrated in Figure 7.2. In the given example, we assume two different states s_0 and s_1. The last known context state is s_1, which has been sent three intervals ago. Since, for both states, the transition matrix p has higher state change probabilities than self-transition probabilities, an alternating state change behaviour is predicted in every step. This is because the NS predictor selects the transition with the highest probability for prediction. As a consequence, the prediction at time t_n is given by state s_2.

Note that for each sampling interval only a single prediction needs to be computed, which can be based on either the last transferred context state or the last prediction (depending on whether an update has been received or not). This leads to a time complexity of $\mathcal{O}(|S|)$ for the calculation. However, as the predictor only considers a

single transition path, the inferred predictions may become inaccurate. For instance, in case the self-transition has the highest probability among all possible transitions for a state s_i (i.e., $\forall s_j \in S, s_j \neq s_i : p(s_i, s_i) > p(s_i, s_j)$), the NS predictor is trapped. In this case, $f(t_{n-k+i}) = s_i$ is always returned as prediction, i.e., the same state s_i is predicted over and over again, although in the real world a user will change his current state eventually again. This problem can be solved by the following predictor that we are proposing.

7.6.2. Multi-Step-Predictor (MS)

In contrast to the NS predictor, the MS predictor is designed to perform a more exhaustive search in order to avoid the problems mentioned above. For this, the MS predictor explores all paths of subsequent transitions for a given prediction horizon, not only the one with the highest probability. More precisely, to predict the context state at time t_n, the predictor determines the conditional probability

$$p(X_{t_n} = s_{t_n} | X_{t_{n-k}} = s_{t_{n-k}})$$

which is given by the k - step transition probability of reaching a state s_{t_n} in exactly k-steps from state $s_{t_{n-k}}$. For the calculation of this probability, all possible transitions are considered that start in $s_{t_{n-k}}$ such that the state is predicted which has the highest probability to be entered over any series of subsequent transitions.

The k-step transition probabilities p_{ij}^k can be computed as the k-th power of p_{ij} (see Section 7.5). The entries p_{ij}^k denote the probabilities for reaching state s_j after taking k steps when starting from state s_i. Based on this information, the prediction can be determined as

$$f(t_n) = s_r \text{ with } r = \arg \max_{j \in \{1,\ldots,n\}} \{p_{idx(s_{t_{n-k}}),j}^k\}$$

i.e., the predicted state is given by the one with the highest k-step transition probability associated with it.

In order to minimize the overhead in calculating such a prediction, the required matrix multiplications for determining p_{ij}^k do not have to be computed all at once. Since the prediction horizon is incrementally expanded, $f(t_{n-k+i})$ is always preceded by the prediction of $f(t_{n-k+i-1})$ for $2 < i \leq k$. For this purpose, the transition matrix p_{ij}^{k-1} can be buffered once it has been calculated, so that p_{ij}^k can be inferred from p_{ij}^{k-1} by requiring only one additional matrix multiplication at any time, i.e., $p_{ij}^k = p_{ij}^{k-1} \cdot p_{ij}$.

In Figure 7.3, we illustrate the principle behind this predictor for the same example as discussed previously. As the figure shows, every possibility sequence of state transitions is considered for the prediction (visually expressed by the grey transition arrows). For a prediction over three time steps, these probabilities can be retrieved from p^3, where

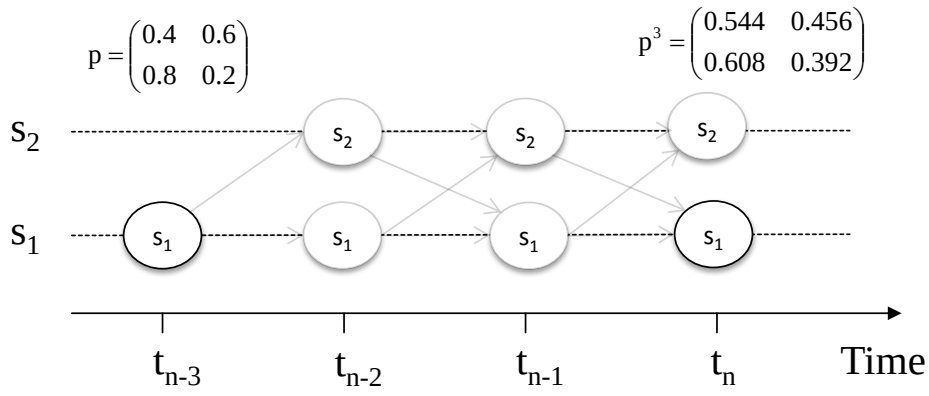


Figure 7.3.: Example of the Multi-Step-Predictor (MS)

the first row in the matrix encodes the probabilities for transitions starting in state s_1 (the last context state received by the consumer). Since according to p^3 the probability to be in s_1 (0.544) after three time steps is higher than the probability to move to s_2 (0.456), s_1 is the result of the prediction. Note that this leads to a different prediction result compared to the NS predictor, where state s_2 has been forecasted.

Due to the incremental approach in calculating p^k, a time complexity of $\mathcal{O}(|S|^3)$ is required for the prediction inference based on the overhead to compute a single matrix multiplication. Compared to the NS predictor, we expect an increased prediction accuracy as the predictor performs a complete search and evaluates all possible state transitions. Also, the prediction cannot be trapped in the same state since the probability to enter a different state may change again with increasing prediction horizons. This eliminates a weakness of the NS predictor in dealing with high self-transition probabilities for predicting a state change.

7.6.3. Expected-Dwell-Time-Predictor (ED)

The ED predictor analyses and predicts the user's dwell time behaviour. The basic idea is that a user spends some limited amount of time in a state before a transition to another state occurs. In order adopt this idea for prediction, two components of a user's behaviour need to be predicted: the time instant at which the next state transition is expected to happen as well as the specific state that will be visited next. Then, a change to the next state is predicted as soon as the estimated dwell time has elapsed, while the previous state is returned until then.

For the prediction of the state that we will be visited next, the transition probabilities are analysed in terms of all other states that can be reached. That is, only states which are different from the last known state $s_{t_{n-k}}$ are considered as possible candidates, since

a proper state change should be predicted. Among these states $s_j \in S \backslash \{s_{t_{n-k}}\}$, the one with the highest transition probability is selected. Hence, the prediction is defined as

$$s_r \text{ with } r = \arg \max_{j \in \{1,\ldots idx(s_{t_{n-k}})-1, idx(s_{t_{n-k}})+1,\ldots,n\}} \{p_{idx(s_{t_{n-k}}),j}\}$$

i.e., the predicted state is given by the one with the highest transition probability which is reachable from $s_{t_{n-k}}$ excluding the self-transition. Note that as a state change is enforced, the prediction cannot be trapped in a single state in contrast to the NS predictor.

Yet, the time at which the transition to s_r will most probably occur is not known. We calculate this time using the dwell time distribution $dw(s_{t_{n-k}})$ associated with state $s_{t_{n-k}}$ that reveals information about the typical duration that a user spends in this state (see Section 7.5). More precisely, we calculate the remaining expected dwell time in state s_i as

$$E_r[dw(s_{t_{n-k}})] = \begin{cases} \lfloor \frac{1}{1-p_{ii}} \rfloor - 1, & \text{if } p_{ii} < 1 \\ +\infty, & \text{if } p_{ii} = 1 \end{cases} \tag{7.5}$$

which indicates the time that is left until the next transition is expected to happen, measured in units of sampling periods. For instance, $E_r[dw(s_{t_{n-k}})]$=2 means that the user is assumed to enter a new state after two sampling intervals, i.e., while the current state persists for one more sampling cycle, the occurrence of a state transition is expected afterwards. The calculation of the expected dwell time is known from the theory of Markov models and depends on the self-transition probabilities based on the formula $E[dw(t_{n-k})] = \frac{1}{1-p_{ii}}$ (see Section 7.5). A high self-transition probability implies that the user spends longer time in a state, since there is only a low chance to leave the state in each cycle. In contrast, a low self-transition suggests that a state change is about to occur soon which leads to short dwell times in turn. In order to compute the remaining expected dwell time, we decrease it by one sampling cycle to account for the fact that the occurrence of a state means that a user has already sojourned in it for one sampling period. Also, we cast the dwell time to an integer value to consider that the predictions needs to be updated in periods of constant sampling periods.

The predictor then uses both predicted information aspects (i.e. predicted state change and expected remaining dwell time) to yield the final prediction, which is defined as:

$$f(t_n) = \begin{cases} s_{t_{n-k}}, & \text{if } t_n < t_{n-k+1+E[dw(s_{t_{n-k}})]} \\ s_r, & \text{if } t_n = t_{n-k+1+E[dw(s_{t_{n-k}})]} \end{cases}$$

As long as the remaining expected dwell time has not elapsed (upper case), the last known state $s_{t_{n-k}}$ is assumed to be still valid. However, as soon as this time is exceeded

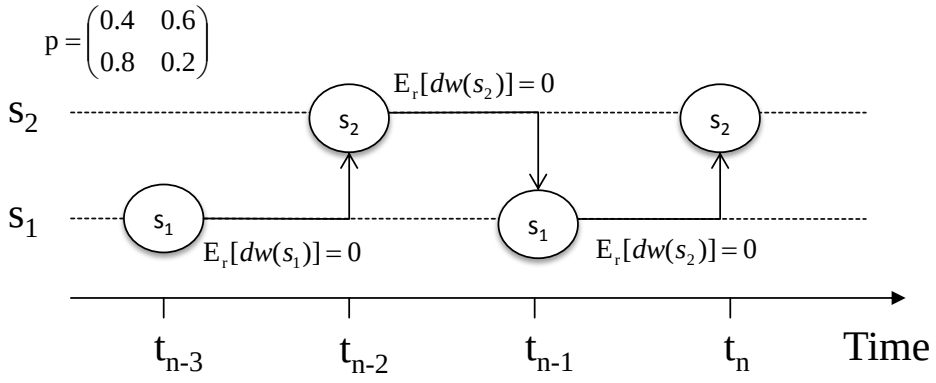


Figure 7.4.: Example of the Expected-Dwell-Time-Predictor (ED)

and the context change is assumed to occur, the transition to the next state s_r is enforced (lower case). The ED predictor applies this scheme continuously as long as no new context update has been received: once s_r has been predicted, the expected dwell time $E_r[dw(s_{t_n})]$ is then calculated for s_r to determine the next state transition and predict the user's behaviour. The complexity for computing the prediction is bound by $\mathcal{O}(|S|)$, as it is dominated by the search for the state successor. The computation of the expected dwell can be done in constant time, since it is linked to the self-transition probability only (see Equation 7.5).

In Figure 7.4, we illustrate the principle behind the ED predictor for our running example. Since we have only two states, the prediction depends on the calculation of the state dwell times, after which a state change is expected to occur. The state dwell times can be derived from the given transition matrix p: Given self-transition probabilities of $p_{11} = 0.4$ and $p_{22} = 0.2$ for states s_1 and s_2, it holds that $E_r[dw(s)] = 0$ for both states (see Equation 7.5) so that a state occurrence is expected to last for one interval only. This results in predictions of alternating state changes, where s_2 occurs at prediction horizon t_n. While this is similar to the NS predictor for our example, there are scenarios where the self-transition probabilities in p cause different predictions. In particular, for self-transition probabilities $p_{ii} > 0.5$, the states are occupied over subsequent time steps as then $E_r[dw(s)] > 0$ holds. These temporal patterns in the user's behaviour cannot be captured by the NS predictor.

7.6.4. Last-Transmitted-Predictor (LT)

Last, we propose the LT predictor that implements a straight-forward prediction scheme. In this case, it is simply assumed that the user constantly remains in the same state which has been reported by the producer. Consequently, no state change has to be considered and the last known state $s_{t_{n-k}}$ is always returned, i.e.,

$$f(t_n) = s_{t_{n-k}}$$

Since no information from the prediction model is required for this approach, the overhead for the calculation of the prediction is low. The time complexity for its computation is constant $\mathcal{O}(1)$. In our evaluation, this predictor is used for the sake of comparison with the more sophisticated predictors. In particular, it allows us to analyse whether knowledge about the state transition dynamics is beneficial for inferring better predictions.

7.7. Evaluation

We have evaluated our approach using real-world data from the CenceMe project [MLF+08], which is available at the Dartmouth database [KH05]. CenceMe is a social networking application that is able to recognize four different activities (S={sitting, standing, walking, running}) of mobile users. The activities have been detected on the users' mobile devices as part of a real-world trial. For the detection of the activities, acceleration data was sensed and classified with a period $\Delta t_s = 8s$. In our evaluation, we have leveraged on the collected traces of user activities to feed a simulator which allows for running update protocols to test mobile sensing scenarios. The simulator is based on a discrete event model to drive a user's behaviour where a new context state is published on the producer in every sensing cycle as recorded in a user's activity trace. The activity traces are also used to train a Markov model which we use as a prediction model for our context predictors as explained in Section 7.5.

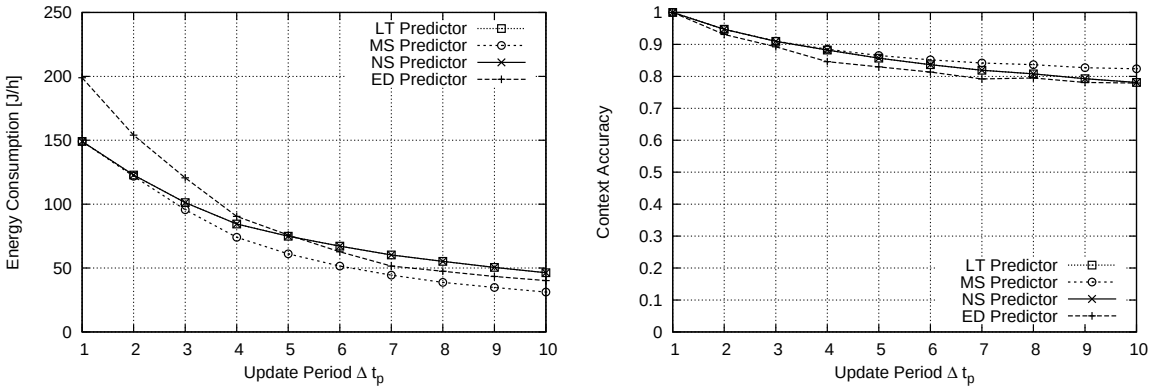
In our evaluation, we analysed the performance of the time-based and deviation-based update protocols (cf. Section 7.4), in combination with the four different context predictors that we developed (cf. Section 7.6). The LT predictor serves as a reference in our evaluation, since it does not require any knowledge from the Markov model to perform the prediction. We have evaluated context accuracy and energy consumption as basic evaluation metrics to reveal the trade-off characteristics of our protocols. Context accuracy is measured as defined in Equation (7.2), revealing the fraction of time the consumer has accurate knowledge about the producer's context state. For quantifying the energy overhead caused by wireless data communication, we rely on an empirical study of GPRS communication on mobile phones [BBV09]. According to the energy model proposed in this study, $e_u = 3.2$J is consumed for a single update operation given update messages of minor size (we assume that context states can be encoded with less overhead so that the message size does not exceed 1KB). This amount of consumed energy arises from the so-called ramp and tail energy to operate the wireless interface in high power mode for data transmission (also see Section 6.3).

In order to allow for a fine-grained performance analysis of our approach, we studied two different evaluation scenarios, where we evaluated the efficiency of our protocols

against synthetic or real traces. For the synthetic traces we learned the activity change patterns inherent to the real trace, but generated new sequences of user activities which adhere to the Markov property. To do this, we sampled from the trained Markov model using a Monte Carlo method, generating random state transitions that are distributed according to the state transition probabilities encoded in the model. As a consequence, the same stochastic process is used for learning and validating the predictions. Moreover, we also validated our protocols against the real traces from the CenceMe data set. By considering both scenarios, we can perform a detailed analysis of possible sources of prediction inaccuracies. On the one hand, we can analyse the theoretical gain of our approach based on the synthetic traces under an accurate prediction model, where prediction errors may only occur as a result from behavioural uncertainties. On the other hand, we can use the real traces to assess the prediction performance in practical settings, where further prediction errors may be caused due to limitations of Markov models in representing real-world behaviour characteristics. This is because a Markov model requires the state dwell times to obey a geometric distribution [How71a]. However, in real traces an arbitrary distribution might appear which may differ from the geometric distribution. Therefore, prediction inaccuracies may arise which are based on the limitation of the stochastic properties of Markov models, rather than the design of the context predictors that we have proposed. A study of these effects allows us to precisely explain the results on which we report in the following.

7.7.1. Synthetic Traces

First, we analysed the time-based update protocols in combination with the four context prediction that we developed. For this analysis, we increased the update period Δt_d as multiples of the sensing interval Δt_s. As Figure 7.5 a) shows, the energy consumption strongly decreases with increasing update intervals. In case of the LT predictor, energy costs of 150J/h are produced for the lowest update interval, while the energy consumption is reduced to approximately 50J/h for the highest update intervals. Hence, the time-based update mechanism turns out to be an effective means of adjusting the energy overhead. For all update intervals, the NS predictor produces exactly the same results as the LT predictor. This is because, in our scenario, we have the highest transition probabilities associated with self-transitions in the Markov model. As a consequence, the NS predictor returns the last transmitted state as prediction. Since by definition the LT predictor always predicts the last transmitted state, both predictors coincide and yield exactly the same forecasts. The MS predictor further improves the performance of the time-based update protocol. On average, the energy consumption is reduced by 18% compared to the NS and LT predictor. The reason is that, in contrast to the NS predictor, the MS predictor performs a complete search over all possible state transitions which enables more accurate predictions. As a consequence, less updates are required to synchronize the states of the producer and consumer. This prediction approach pays off especially for longer prediction horizons (i.e. in case of higher update

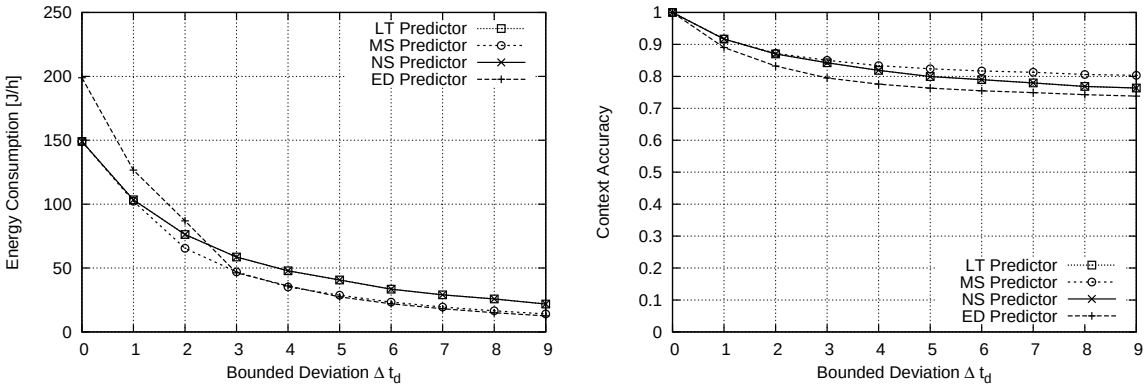


(a) Energy Consumption (b) Context Accuracy

Figure 7.5.: Synthetic traces: time-based update protocol

intervals), where often a change in the user' activity needs to be predicted. While the NS predictor always forecasts the last transmitted state in this case, the MS predictor is able to determine the user's new activity so that no update message is required for the consumer to obtain this information. The ED predictor shows a weak performance for update intervals $\Delta t_p < 4 \cdot t_s$. In our evaluation, we observed that there is a high variance in the state dwell times, which makes an accurate estimate of the remaining dwell time in a state hard to predict. In particular, if a state change is predicted to happen earlier than it actually occurs, additional updates are necessary to correct the false predictions which negatively affects the performance. This is especially critical for lower update intervals, where already small errors in the predicted dwell times can cause additional updates. Therefore, we conclude that the ED predictor is efficient for higher update intervals only.

The impact on the context accuracy is plotted in Figure 7.5 b). As can be seen, the context accuracy does not significantly suffer from higher update intervals where the update message rate is reduced. We can observe that the context accuracy drops from 1 for the lowest update interval to approximately 0.8 for the highest update interval. This behaviour is similar across all protocol variants independent of the specific prediction approach due to the time-based update decision. The time-based update mechanism ensures that context changes with a high occurrence probability are updated regularly. These are exactly the updates which create the most positive effect on the context accuracy. While short changes in user context may remain unnoticed with higher update intervals, states which are occupied over a longer time are transmitted when the next update is scheduled. The LS predictor and the NS predictor achieve exactly the same accuracy since they are based on the same predictions in our scenario, as explained before. The MS predictor is able to achieve a slightly better accuracy since the context states of producer and consumer match more often due to the more sophisticated prediction strategy. The performance of the ED predictor is slightly worse as state



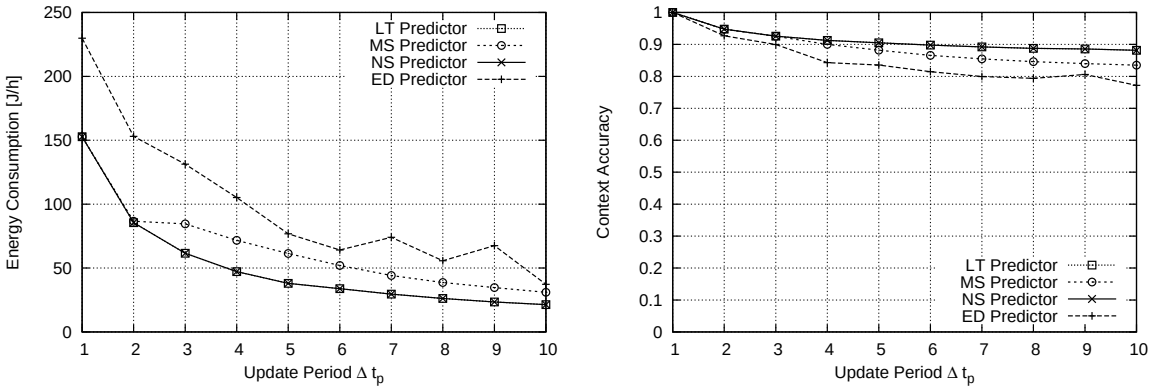
(a) Energy Consumption (b) Context Accuracy

Figure 7.6.: Synthetic traces: deviation-based update protocol

changes are often predicted too early or too late due to the high variance in state dwell times. We conclude that the context accuracy can be preserved very well despite of the strongly reduced energy consumption. For instance, in case of the MS predictor, a significant accuracy of 0.85 can be achieved if only 35% of the energy is spent that is required to achieve perfect accuracy.

In Figure 7.6 a), we analyse the energy consumption for the deviation-based protocol, where the deviation period Δt_d is increased as multiples of Δt_s. In qualitative terms, the results confirm our measurements that we gained for the time-based protocol. The MS predictor is superior to all other predictors and achieves a significant reduction in energy consumption by 23% on average. The ED predictor provides improvements for higher deviation bounds only, as variances in predicted dwell times can be much better compensated in this case. The NS predictor and LT predictor behave exactly in the same way due to the high self-transition probabilities. In quantitative terms, however, higher energy savings can be achieved by the deviation-based protocol compared to the time-based protocol. This is because a less strict update criterion is used, which tolerates short-time deviations of context changes that are not reported to a consumer.

The effect of the deviation-based protocol on the context accuracy is illustrated in Figure 7.6 b). Compared to the time-based protocol, the context accuracy is slightly worse. This is because less updates are sent with the deviation-based protocol so that consumers have less accurate knowledge of the user's actual context state. As a consequence, the uncertainty at the consumer further increases and also the predictions become less accurate since they need to be based on updates which have been sent quite some time ago. However, the trade-off between energy consumption and context accuracy is better with this approach: For instance, using the MS predictor, a significant accuracy level of 0.85 can be achieved when consuming only 32% of the total energy that is required for transferring every context change.



(a) Energy Consumption (b) Context Accuracy

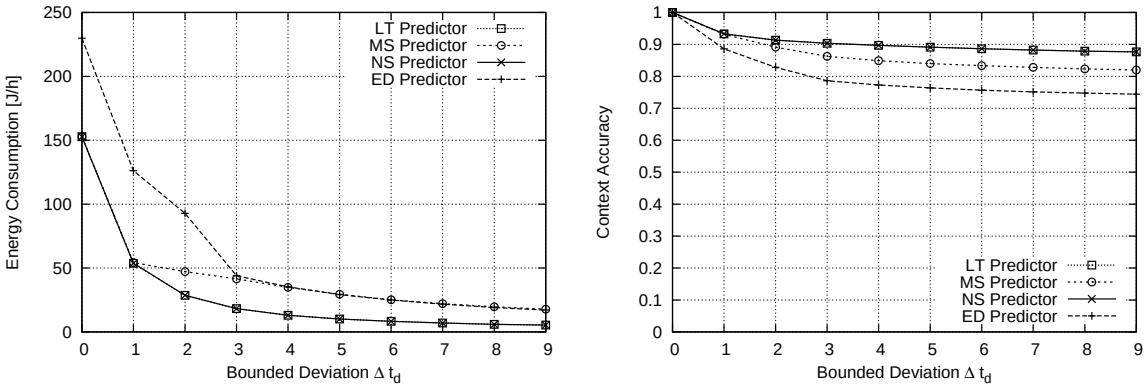
Figure 7.7.: Real traces: time-based update protocol

7.7.2. Real Traces

In the following, we report on the evaluation results we have gained for the real traces. As we will see, the evaluation reveals some inherent limitations in modeling real-world user behaviour with Markov models. Subsequently, we will analyse these problems in detail, and then design a new prediction-based update protocol that circumvents the found drawbacks in the next chapter.

First, we analyse the energy consumption required by the different approaches. As shown in Figures 7.7 a) and 7.8 a), the MS and the ED predictor consume more energy than the NS and LT predictor. This is in contrast to the results gained from the synthetic traces reported in the last section, for which especially the MS predictor exhibited a good performance. Due to the high-self transition probabilities in the given user trace, the NS and LT predictors result in forecasts which assume the user to remain in the state that has been reported last. In contrast, the MS and the ED predictor denote more sophisticated predictors which reason over longer-term user behaviour to anticipate actual state changes. For the real trace, predicting actual state changes seems to be aggravated, producing inaccurate context state information at a consumer which causes an increased message overhead for synchronization. The reason is that, even though we train our Markov model with the real-world context traces, the user's behaviour in the given scenario does not satisfy certain statistical assumptions that are required for a discrete Markov model to be an effective predictor. More precisely, a Markov model, as a representation of a user's behaviour, models state dwell times implicitly using transition probabilities and therefore requires the dwell times to be distributed according to a geometric distribution [How71a](see also Section 7.5).

However, for the studied real-world traces this assumption is not given. This is demonstrated in Figure 7.9 which plots the distribution of dwell times in the user state 'running'. On the one hand, the figure shows the actual distribution of dwell times as



(a) Energy Consumption (b) Context Accuracy

Figure 7.8.: Real traces: deviation-based update protocol

found in the real trace. On other hand, a geometric distribution is plotted which is assumed by the Markov model and can be analytically determined using the transition probabilities (cf. Section 7.5). If the real-world behaviour would be reflected accurately in the Markov model, both distributions should closely match. However, as can be seen, the geometric distribution assumed by the Markov model significantly deviates form the real-world user behaviour characteristics. As a consequence, the transition probabilities stored in the Markov model imply sequences of context changes which do not occur with the same distribution in the traces of the user's behaviour.

Note that this problem is caused by the fundamental statistical properties of a Markov model, and not the learning and prediction method underlying our approach. As a consequence, the predicted context changes are often inaccurate and need to be corrected, consuming a substantial amount of energy for required additional update messages on the producer. The same observation can also be made for the context accuracy shown in Figures 7.7 b) and 7.8 b). The best accuracy is provided by the NS and LT predictors, while the MS and the ED predictors yield worse results. This demonstrates that frequently context state transitions are predicted (MS and ED predictor) while the user has remained in his previous state (NS and LT predictor). Consequently, the sophisticated predictors MS and ED cannot be effectively applied to our scenario as they are susceptible to inaccurate patterns stored in the Markov model. Actually, they are outperformed by the simple predictors which are able to achieve excellent energy/accuracy trade-off characteristics. In case of the time-based protocol, 89% accuracy can be achieved with only 22% of the energy which would be required for providing perfect accuracy. Even better, in case of the deviation-based protocol, for the same accuracy level only 7% of the energy is required. Since in the real-world user trace a state is occupied only for very short time (see Figure 7.9), a substantial amount of energy can be saved with relaxed update criteria (i.e. higher update interval or deviation threshold) while the context accuracy remains at a high level.

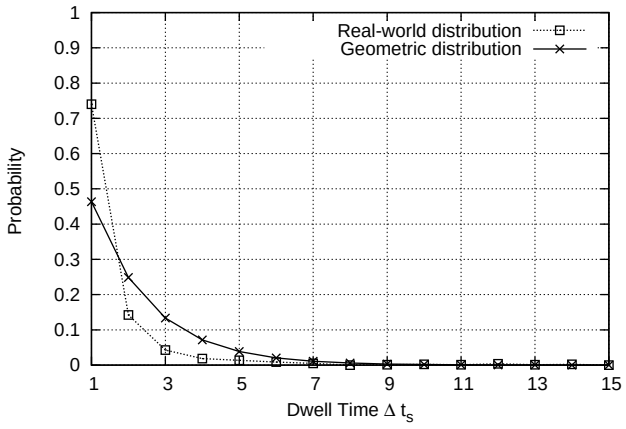


Figure 7.9.: Dwell time distribution

7.7.3. Discussion

Having designed update protocols to improve the trade-off between energy consumption and context accuracy, we could analyse in our evaluation the extent to which context predictions can help to increase the effectiveness of the protocols. The evaluation over synthetic and real-world traces has further revealed issues related to the statistical properties of Markov models which affect our predictors. Altogether, the following factors have been encountered:

1. In order to reduce the message overhead for context updates, context state transitions need to be predictable with a high accuracy. Due to the natural variances in human behaviour, accurate predictions of the exact transition times are hard to achieve especially for low deviation bounds or update intervals.
2. In addition, we have seen that the Markov model based prediction scheme is vulnerable to statistical deviations in the state dwell times. While a Markov model requires the state dwell times to be distributed according to a geometric distribution, they may follow an arbitrary distribution in real traces.
3. While our protocols aim at improving the energy/accuracy trade-off, they are not optimised towards a maximum allowance of energy which can be afforded by the application. In order to make optimal usage of the energy available on a device, update protocols would have to adapt to the rate of context changes in the user trace instead of applying fixed thresholds or update times.

The insights gained from our evaluation require us to revisit our approach in how context predictions should be exploited best to enhance update protocols for discrete user context. To this end, we present an enhanced protocol in the next chapter which is based on a more effective prediction strategy. Instead of aiming at a best-effort improvement of the energy/accuracy trade-off, the goal of the update protocol is to the

maximize the context accuracy at a consumer under a given energy budget (formulated as a constrained optimization problem). The design of a protocol which addresses this problem allow us tackle the three issues discussed above that are key for a good protocol behaviour.

First, our revised prediction strategy does not require strictly accurate forecasts of context state transitions to reduce the number of updates. Instead, we predict which updates would be most effective to maximize the context accuracy at a consumer for a given energy limit. As context predictions are used to inform the prioritization of updates only, no additional update messages are required to correct inaccurate predictions. Second, with our revised strategy, state dwell times are used to compare dwell behaviours among different states. An exact prediction of the time when a context transition is supposed to occur is not relevant for this approach. As a consequence, the prediction strategy is more robust towards deviations of the Markov model from real-world behaviour patterns. Third, the optimisation takes a limit on the consumable energy as an input parameter. The rate of expected changes is predicted and used to decide if sending updates would exceed the energy budget. By adapting the update decisions to the patterns inherent to a user's context changes, the available energy can be used optimally. Hence, the insights gained in this chapter can be leveraged to inform the design of an enhanced prediction-based update protocol which is more robust and has more practical relevance for real-world scenarios.

7.8. Related Work

The design of efficient update protocols has received a lot of attention in the field of mobile computing in the past. Over the last decades, update protocols have been developed that are tailored towards the specific purposes of mobile systems.

In the first generation, the focus has been on update strategies for cellular wireless networks [BD99]. In order to establish mobile phone calls, the cell-based user locations have to be known to the cellular network provider. Basically, two different approaches can be used to learn about the users' locations in the cellular network: querying for the current network cell in which a user is located using paging messages, and reporting about changes in the location of a user using update messages [RMD07]. In terms of the overhead imposed on the cellular network, there is an inherent trade-off between update and paging messages, since the number of required paging messages depends on the uncertainty about a user's location. In order to find a good balance between both, three strategies have been analysed in terms of their trade-off characteristics [BNKS94]: a time-based, a movement-based, as well as a distance-based strategy. For reasoning over the characteristics of these protocols, the cell-based layout of the wireless network must be considered: As long as the mobile device could not be found at the cell that was reported last, the search needs to be gradually expanded along the neighbourhood

of adjacent cells. Hence, the proposed protocols are designed to locate users in cellular wireless networks, where the cell-based distance of the user's current location from the last reported location has a big impact on the resulting paging cost. As a consequence, these protocols do not translate to the purpose of our mobile systems where context states can represent any form of semantic information about the users and the context accuracy at interested consumers is of primary interest.

With the proliferation of position technology on mobile devices, the second generation of update protocols focused on tracking of users' fine-grained locations. These locations are described as geographic positions relative to a reference coordinate system, such as the World Geodetic System 1984 (WGS84) [WGS94] used by the Global Positioning System (GPS). In order to achieve scalable location data management, the users' positions are stored at location servers according to a spatial accuracy bound. The accuracy bound is a configuration parameter which guarantees that the positions published by the location servers do not deviate from the true positions by more than a given maximum distance. Leonhardi et al. have presented a comprehensive overview of update protocols for transferring position data from mobile devices to location servers [LR01]. Specifically, time-based and distance-based update protocols are discussed that make use of characteristic properties of moving objects such as their maximum speed to defer update messages by a maximum delay that is guaranteed to satisfy the accuracy requirements. Moreover, a prediction-based update protocol has been presented by Wolfson et al. [WSCY99] using the concept of dead-reckoning. With dead-reckoning, a prediction function is reported to the location server so that updates are only required for positions which deviate too far from the predictions. The most simple prediction is a linear function given by the future velocity and direction of movement. In contrast to geographic positions, we focus on discrete context data (e.g. activity information such as running) in our work, where assumptions about user mobility in the geographic space do not hold. In order to support context updates under a discrete context model, new approaches are required for measuring context accuracy and making context predictions. For instance, a linear function cannot be applied to implement a prediction-based update mechanism for context states. Instead, an approach is required which reasons over the probabilities of transitioning among different context states. Furthermore, a discrete context model requires a specific measure of context accuracy, which needs to be reflected in the update criteria on which update protocols are based. In our approach, we measure context accuracy as the deviation of two context states using a boolean metric. Hence, the underlying context model has far-reaching implications on the protocol-specific measures and prediction techniques that can be employed.

Discrete context is in the focus of new mobile applications, which continuously sense context states on mobile devices and share them with back-end servers in real-time. In order to help this emerging field to proliferate, new update protocols are required to deal with the trade-off between message overhead and context accuracy. Musolesi et al. have presented various protocols to deliver an efficient uploading process for discrete

context states [MPF+10]. The key idea of these protocols is to report only context changes that remain valid over a significant period of time. However, this approach does not allow for a fine-grained control of the message overhead and its impact on the context accuracy for a wide range of different scenarios. In the worst case, the context accuracy may be even near zero in scenarios where persistent context changes do not frequently occur so that no update message is triggered. In contrast, we allow with our protocols to limit the deviation of the information at a consumer from the real context. Both the time-based and the deviation-based protocol can be configured with time intervals to control the trade-off between message overhead and context accuracy in an effective manner. Further, we use predictions to support the uploading process in an online-modus, where updates shall be received in real-time. For this purpose, we have designed various predictors that can infer accurate predictions of future context states at distant times. The development of sophisticated context predictors for discrete context data which can be integrated into context update protocols has not been considered by previous work. Also, our evaluation gives insight in the energy consumption caused by context updates, which has not been studied in [MPF+10]. Consequently, we can conclude that we have performed a comprehensive study of prediction-based protocols for efficient updates of context states which has not be seen in previous research.

7.9. Summary

In this chapter, we have investigated strategies for the energy-efficient distribution of discrete context data (e.g. user activities) in mobile sensing applications. In these applications, the users' context is captured on their mobile devices for delivering context updates in real-time to a large number of interested consumers (e.g. friends in an online social network). Since frequently sending updates messages over wireless communication channels is energy intensive and can quickly drain the batteries of mobile devices, we have proposed a set of prediction-based update protocols that are characterized by an inherent trade-off between energy consumption and context accuracy. For this purpose, we have presented two basic update protocols, a time-based and a deviation-based update protocol, and four different context predictors that can be exploited to predict missing context updates. The proposed context predictors reason over the state transition probabilities recorded in Markov models to infer suppressed context updates on a consumer based on the last received update message. By integrating the context prediction scheme into our update protocols, we can avoid the transmission of costly update messages and achieve an effective control of the trade-off between energy consumption and context accuracy for various scenarios.

In our evaluation, we have analysed the performance of our approach based on a real-world case study. We have found that in case the user behaviour obeys the Markov property, our protocols can achieve significant improvements of the energy/accuracy trade-off. However, our evaluations have also shown that the protocols are susceptible

to non-Markovian real-world behaviour where the statistical assumptions about the dwell time distribution are violated. Having further analysed this problem in detail, we have elicited requirements for an improvement of our approach towards a more robust protocol. Specifically, the idea of using context predictions to select the most important context updates under strict energy bounds has been discussed to yield a more effective protocol. These findings are incorporated into a new prediction-based update protocol which is presented in the next chapter.

Chapter 8

Predictive Context Update Protocols with Hard Energy Constraints

8.1. Introduction

In this chapter, we present a rigorous approach to support the energy-efficient execution of mobile sensing applications. For this purpose, we revisit and expand our approach presented in the last chapter to optimise the context update process on mobile devices. The underlying rationale is that mobile sensing applications should be strictly controlled in terms of their maximum energy rate to give hard guarantees about the usability of mobile devices. Recent studies have shown that a significant amount of background energy is consumed on mobile devices in everyday scenarios caused by active screens, voice calls, etc. [FMK+10]. Since users are not willing to give up services such as telephony or video playback that have become indispensable in their daily lives, a fundamental energy bottleneck is created which limits the amount of available energy to further applications. Novel mobile sensing systems will therefore only be tolerated by end users if they do not impede the everyday usage of their mobile devices. Since mobile sensing applications rely on frequent sensing and communication operations that cause a substantial energy overhead [RZ07], hard guarantees about the energy consumption are required to enable a wide acceptance of these applications among users.

In prior research, various approaches have been proposed to reduce the energy consumption on mobile devices both in terms of sensing a user's context as well as communicating the context information to remote parties [MPF+10, KLGT09]. For instance, in order to lower the wireless communication overhead, it has been shown that not uploading short-term context changes can yield substantial energy savings [MPF+10]. Similarly, it has been proposed to suspend the location sensing process on a mobile device in situations where the user is detected to be not moving [KLGT09]. However, these approaches are based on heuristics which can be applied opportunistically only and

the extent to which these conditions hold cannot be controlled. As a consequence, the devices' batteries may be drained in an unforeseen manner and an upper bound on the energy usage cannot be provided. Therefore, we propose to take a more rigid approach by restricting the energy usage of a mobile applications by means of contractual design. As part of this contract, a limited energy budget is made available to mobile sensing applications that provides the amount of energy that may be consumed for sensing and communication of context data. Given hard bounds on the allowed energy consumption, a particular research challenge which arises in this context is how to exploit the available energy to achieve the best quality of service delivered by these applications. In particular, since constantly reporting about context updates is prohibited due to the high energy costs of sensing and communication operations, the available energy should be dedicated to those operations which can maximize the context accuracy experienced by interested consumers.

In order to address this challenge, we propose in this chapter novel context update protocols that exploit knowledge of the user's future behaviour to decide for the most effective context updates under a given a limited energy budget. For the development of these protocols, we present an optimization approach which is based on a stochastic decision process (Constrained Markov Decision Processes) to find the optimal protocol configuration. The decision process is employed to model the energy costs and expected accuracy which would result from enforcing context updates in different user states with a certain probability (i.e. whether to sense and transmit a context state). Predictions of the future user context are incorporated into this process to provide accurate cost and accuracy estimations incorporating the user's prospective behaviour. Solving the decision process for a specific user yields the optimal configuration with the maximum context accuracy that can be achieved under a given energy budget.

We show how to leverage on this approach to design update protocols for two different scenarios. First, we only consider communication costs and derive the optimal uploading policy for transmitting context updates given energy bounds on the wireless communication overhead. Then, we additionally consider sensing costs and extend our approach to infer an effective sensing and communication schedule which respects bounds on the total energy consumption on a mobile device. In our evaluation, we demonstrate for a real-world context trace from a mobile social networking application that our predictive protocols are far more effective than protocols that ignore knowledge of future user behaviour. As a result, we have developed a generic solution which can be used whenever streams of discrete context data need to distributed under a given energy budget, supporting the development of effective mobile sensing applications for a wide range of scenarios.

The rest of this chapter is organized as follows. In Section 8.2, we develop novel context update protocols to report context changes over wireless networks with hard energy bounds. In Section, 8.3, we then describe an extension of this approach and propose a sensing-and-update protocol which is able to respect bounds on both the sensing and

communication costs. In Section 8.4, we present a comprehensive evaluation of our update protocols for real-word context traces where we compare the performance of our optimizations with two baseline protocols. Finally, we conclude this chapter with a summary in Section 8.6.

8.2. Predictive Update Protocol with Hard Energy Bounds

In this section, we present a predictive update protocol that respects hard bounds for the energy consumed by context updates sent over wireless communication channels. First, we formalize the problem of maximizing the context accuracy under a given energy budget as a constrained optimization problem. Based on this formal problem statement, we then describe different strategies and algorithms for solving this problem.

8.2.1. Problem Statement

In order to inform consumers about sensed context changes, context updates over wireless cellular radio channels (e.g. GPRS) are required. As defined in Section 7.2, let e_u denote the energy required for a single update message. To control the communication overhead on a mobile device, we make only a limited amount of energy available which constrains the number of update messages that can be sent. Formally, this energy budget is denoted as E, which should be carefully chosen to guarantee a sufficient lifetime of the device's batteries. Therefore, a protocol is required that guarantees that the energy consumption on a mobile device does not exceed E.

To avoid exceeding E and ensure the usability of the device, certain context updates may have to be suppressed. The challenge is to select the most effective updates $U \subseteq T$ for transmission, while the remaining updates $T \setminus U$ are omitted. The choice of $U \subseteq T$ has a major impact on the accuracy of the context perceived by a consumer. The context $c(t)$ known to the consumer at time $t \in T$ is given by the most recent update from the producer, i.e., $c(t) = p(t')$ with $t' = \max\{u \in U | u \leq t\}$. If a context change is not transmitted, then the consumer and producer context may differ ($c(t_i) \neq p(t_i)$), resulting in inaccurate context information at the consumer. Therefore, the effect of suppressing context updates has to be minimized and the most effective updates have to be found.

The accuracy of the context at a consumer is measured by a basic accuracy function. This function is defined as

$$\lambda(t) = \begin{cases} 1, & \text{if } p(t) = c(t) \\ 0, & \text{else} \end{cases} \tag{8.1}$$

that yields whether the producer and consumer states match at time t. Specific applications may have distinct interests in certain context states, i.e., certain states

may be considered as highly relevant, while other may be not as important. Therefore, the accuracy measure should reflect how well the interests of a consumer are satisfied. Taking this into account, we define the accuracy as

$$\lambda_w(t) = w(p(t)) \cdot \lambda(t) \tag{8.2}$$

where $w(s_i) \in [0,1]$ (or w_i in short) denotes an application-specific state weight for $s_i \in S$. Thus, we include a measure of state relevance in the basic accuracy function. Normalizing (8.2) over the application lifetime $|T|$ results in

$$ac = \frac{\sum_{t=1}^{n} \lambda_w(t)}{|T|} \tag{8.3}$$

as a basic measure for the consumer-side context accuracy, indicating the fraction of time for which the consumer is in the correct state as a value in $[0,1]$, weighted by the state relevance factor. Hence, to provide consumers an accurate view on the producer's context, high values of ac are essential.

Formally, the goal is to find the optimal set of updates times U such that the following constrained optimization problem is solved:

$$\text{maximize} \qquad \frac{\sum_{t \in T} \lambda_w(t)}{|T|} \tag{8.4}$$

$$\text{subject to} \qquad \frac{\sum_{t \in T} \mu(t)}{|T| \cdot \Delta t_s} \leq E \tag{8.5}$$

The optimization goal (8.4) is to achieve the maximum accuracy ac of the context perceived by the consumer. The optimization constraint (8.5) limits the energy consumption ec of the communication overhead by E (cf. Equation (7.4) where ec has been introduced). In the following sections, we present a novel approach to address this problem. Since the maximization of the context accuracy depends on the chosen context updates, the prospective user behaviour will have a major impact on the effectiveness of a context update. Therefore, a prediction-based approach will be taken to forecast the future user behaviour and estimate the expected energy costs to decide whether the energy budget permits a context update to be sent or not.

8.2.2. Approach Overview

An overview of our approach is shown in Figure 8.1. First, we learn a stochastic model of a user's context changes. The model is based on a discrete Markov model and gives important insight into the future user behaviour (e.g. the expected time to stay in a particular context state) (cf. Section 7.5). Then, we apply a stochastic optimization

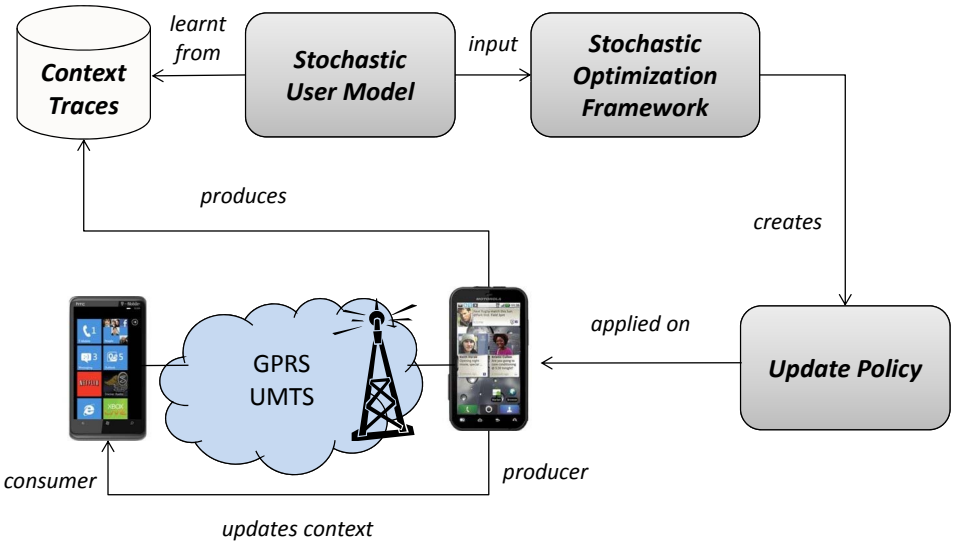


Figure 8.1.: Approach for configuring our predictive update protocol based on a stochastic optimization framework

framework to derive an update policy for a given energy budget. Note that this marks an important difference to the approach presented in the last chapter, where the energy budget has not been taken into account a priori to the configuration of the update protocol. As we consider the energy budget as part of the optimization, our update protocols are now targeted towards a maximum energy utilization. The optimization framework uses the Markov model to find an update policy which guarantees the best selection of U.

The stochastic update policy gained from the optimization framework is used to control the update process on a mobile device based on probabilistic decisions. More precisely, each update is taken with a probability which is related to the predicted gain in consumer-side context accuracy. With high probability, we prioritize updates of relevant context states, in which the user is often found after the update (as a result of staying in the state or caused by re-visiting the state). The rationale behind this is that transmitting such a state results in the consumer having accurate context information of a relevant state for a longer time on average, resulting in a high accuracy measure (cf. Equation 8.2). This means that an update is particularly effective if it matches the prospective behaviour of the user for most of the time, until the next update is made. Consequently, by means of the update probabilities, the energy overhead as well as the impact on the context accuracy can be effectively controlled.

While the update policy is applied on the mobile device, our approach is flexible as to where the stochastic optimization framework is executed for computing the policy. Since on the mobile device only the result of the stochastic optimization process is required, the inference can be done a priori on a server which can easily deal with the underlying

computational overhead. We investigate two alternatives of stochastic optimization, which differ in their complexity and exactness of modeling our problem. We consider a *memory-less update protocol* where only the current context change of the producer is relevant for the update decision, and a so-called *update protocol with memory* which also incorporates the last context state that was reported to the consumer before. Both variants are explained in detail in Sections 8.2.3.1 and 8.2.3.2. The update policies may then be recomputed regularly to compensate for changing user behaviour. In the following sections, our focus is on how to compute the update policies.

8.2.3. Update Protocols

In the following, we propose a prediction-based update strategy to optimize the update decisions. For this purpose, we present two different protocol variants to exploit the knowledge from the Markov model for computing predictions, as explained subsequently.

8.2.3.1. Memory-less Update Protocol

The memory-less predictive update protocol (PreUP-Memless) follows a simple idea to restrict the number of updates messages on a device. For each context change that is observed locally, a selective update decision is taken to trigger or deny a transmission. Formally, the update decision is given by a policy $u : S \rightarrow [0, 1]$ that returns the probability $u(s_i)$ (or more compactly u_i) for sending an update once a context change to state $s_i \in S$ has been observed. The policy is then used as the basis for a probabilistic experiment with binary outcome, where a context change is only transferred to a consumer if the experiment results in a positive update decision. The update protocol is applied over the entire application lifetime t_{max} to control the update process. We assume that t_{max} refers to the duration of a typical recharge cycle. Due to the probabilistic update decision u_i, an update of state s_i can happen at any time over the application lifetime.

The policy u is applied on the mobile device of the producer using Algorithm 7. Periodically, the producer samples the current context s_C (line 3). If the application has just started, no transmission has yet taken place (line 4). Therefore, we send an initial context update (lines 5-7). Further context updates are only triggered by observed context changes. If such a context change occurs (line 8), the update policy is used to enforce the update decision. For this purpose, a probabilistic decision is made that is dependent on the update probability $u(s_C)$ associated with state s_C (line 9). If the update policy triggers an update (line 10), s_C is transferred to the consumer, and the remaining energy budget shrinks by e_u (lines 11-12). Finally, we store the current context in s_L to discover a possible context change in the next cycle (line 14). Even though the update policy has been chosen to limit the energy consumption rate by E, the available energy budget may be drained at some point in time due to deviations

Algorithm 7 Memory-less Update Protocol

```
1: s_L ← null
2: while E · t_max ≥ 2 · e_u do
3:     s_C ← retrieve current context
4:     if s_L = null then
5:         send update message(s_C)
6:         E ← E − e_u
7:         s_L ← s_C
8:     else if s_C ≠ s_L then
9:         r ← rnd()
10:        if r ≤ u(s_C) then
11:            send update message(s_C)
12:            E ← E − e_u
13:        end if
14:        s_L ← s_C
15:    end if
16:    sleep(Δt_s)
17: end while
18: s_M ← s_r with r = arg max_{i∈{1,...,n}}{π_i}
19: send update message(s_M)
```

of the predicted from the actual behaviour. Therefore, the energy consumption is continuously monitored and in case only energy for a single update message is left, we transfer the most likely state in a final update message. This is the context state with the highest stationary probability (see Section 7.5), so as to maximize the context accuracy for the rest of the application's lifetime (line 18).

In order to find an efficient update policy u, we apply an optimization technique where we model the effects of the update decisions as functions which depend on the assignment of update probabilities u_i, $\forall s_i \in S$. Specifically, we propose two functions to estimate the expected energy costs and the expected context accuracy for a policy u. Beyond the update probabilities which need to be chosen, no further unknown variables exist since we can rely on the Markov model as an expectation of the user's future behaviour. As a result, we can provide our functions as input to a problem solver, which can use the knowledge about a user's context change patterns to maximize the context accuracy under a given energy constraint. As we will see, we can use Linear Programming as optimization technique since both functions are linear dependent on the update probabilities u_i.

For estimating the expected accuracy $E_a(u)$, we try to determine the gain in accuracy by predicting how long a context state will be valid after an update. For this purpose, we use the expected dwell time as available from the Markov model to forecast how long

a user will remain in a state before leaving it again. Therefore, the expected accuracy $E_a(u)$ of the consumer can be expressed as follows:

$$E_a(u) = \sum_{s_i \in S} \pi_i \cdot \Big(\sum_{s_j \in S : s_j \neq s_i} p_{ij} \cdot u_j \cdot \frac{1}{(1 - p_{jj})} \cdot w_j \Big) \tag{8.6}$$

$E_a(u)$ is composed of different predictions. First, we determine the joint probability of the occurrence of context state s_i and a subsequent transition from s_i to s_j ($s_i \neq s_j$). This probability is given by $\pi_i \cdot p_{ij}$, where π_i denotes the stationary probability of state s_i and p_{ij} is the transition probability (see Section 7.5). This probability needs to be multiplied by $u(s_j)$ to determine the likelihood that an update is sent when a transition from s_i to s_j occurs. The gain we receive from sending an update depends on the expected dwell time (given by $(\frac{1}{(1-p_{jj})})$ as discussed in Section 7.5), which is an estimation of the time that the producer stays in the updated state before the next change occurs. For each context state, we also have to take the associated state relevance weight w_j into account. $E_a(u)$ is thus the prediction of the consumer-side accuracy λ_w (cf. Equation 8.3), expressed as the expected probability of knowing the correct context state for a given update policy u.

The expected energy cost $E_c(u)$ which result from a specific assignment of update probabilities can be computed in a similar way. It is given by:

$$E_c(u) = \Big(\sum_{s_i \in S} \pi_i \cdot \Big(\sum_{s_j \in S : s_j \neq s_i} p_{ij} \cdot u_j \Big) \Big) \cdot \frac{e_u}{\Delta t_s} \tag{8.7}$$

To quantify the expected energy demand, e_u (the energy required to transmit a single context update) is multiplied by the probability that an update message is sent. Since such an update may happen only in periods of sampling intervals, we divide the amount of required energy by Δt_s to compute the average energy consumption per time. Consequently, $E_c(u)$ is the energy consumed per time for transmitting all context updates (cf. ec as defined in Equation 7.4) under the policy u.

We can observe that both $E_c(u)$ and $E_a(u)$ are linearly dependent on u. Therefore, we apply Linear Programming (LP) as an optimization method to find the best assignment of update probabilities which can maximize the resulting context accuracy, while not exceeding the available energy budget. The LP representation of our problem is given by:

$$\text{maximize} \qquad \sum_{s_i \in S} \sum_{s_j \in S : s_j \neq s_i} \pi_j \cdot p_{ji} \cdot u(s_i) \cdot \frac{1}{1 - p_{ii}} \tag{8.8}$$

$$\text{subject to} \qquad \sum_{s_i \in S} \sum_{s_j \in S : s_j \neq s_i} \pi_j \cdot p_{ji} \cdot u(s_i) \cdot \frac{e_u}{\Delta_{t_s}} \leq E \tag{8.9}$$

$$\text{and} \qquad 0 \leq u(s_i) \leq 1, \forall s_i \in S \tag{8.10}$$

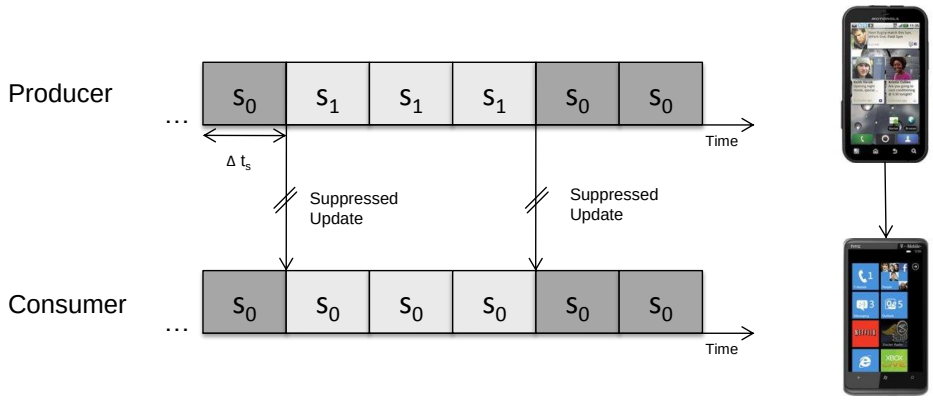


Figure 8.2.: Example of an accurate context despite of missed updates

Equations (8.8) and (8.9) are transformations of (8.6) and (8.7), such that each unknown variable $u(s_i)$ appears once in the equation. The constraint in (8.10) is necessary to arrive at a valid probability measure for the update policy, so that all update probabilities satisfy $0 \leq u(s_i) \leq 1$. Solving the LP problem then yields the solution for our memory-less update protocol. We have used a state-of-the-art problem solver for this purpose which is available in MATLAB.

8.2.3.2. Update Protocol with Memory

The memory-less approach presented in the previous section assumes that the context accuracy at a consumer is degraded with each suppressed context update. However, we can observe that the producer and consumer context might match even when no update is sent. This may happen if consecutive updates are suppressed, and the producer's current context returns to the state of the last update. An example is given in Figure 8.2, where the producer's context first changes from s_0 to s_1. As the update is suppressed, the consumer will assume an incorrect context as a result. However, because the producer enters s_0 again later, both states match again. In this situation, an explicit update for the change from s_1 to s_0 is not required to increase the context accuracy. This observation can be exploited to further improve our update protocol by allocating energy only to those updates that create a positive impact on the context accuracy.

Therefore, we propose the predictive update protocol with memory (PreUP-Mem) that is able to incorporate the most recently transmitted context into the update decision. This will avoid situations where an update is sent that is already known to the consumer to further increase the effectiveness of our protocol. Formally, we apply an update policy r, which is defined as a function $r : (S \times S) \rightarrow [0, 1]$. The policy returns the probability $r(s_j, s_i)$ (in short $r_{j,i}$) for sending an update of the current context state s_i given the last transmitted context state s_j. We use r to control the transmission of

Algorithm 8 Update Protocol With Memory

```
1: s_U ← null
2: while E · t_max ≥ 2 · e_u do
3:     s_C ← retrieve current context
4:     if s_U = null then
5:         send update message(s_C)
6:         E ← E − e_u
7:         s_U ← s_C
8:     else
9:         r ← rnd()
10:        if r ≤ r(s_U, s_C) then
11:            send update message(s_C)
12:            E ← E − e_u
13:            s_U ← s_C
14:        end if
15:    end if
16:    sleep(Δt_s)
17: end while
18: s_M ← s_r with r = arg max_{i∈{1,...,n}}{π_i}
19: send update message(s_M)
```

context updates as shown in Algorithm 8. The algorithm requires to store the most recently transmitted update in variable s_U. We initialize s_U on application start-up (line 7) and each time an update is sent (line 13). In our protocol, each context occurrence may potentially trigger an update message, i.e., even if two sensed consecutive context states are equal ($s_{t_i} = s_{t_{i+1}}$), the update policy is evaluated to make a decision whether an update should be sent (lines 9-14). This is done to consider the fact that due to low update probabilities the state $s_{t_{i+1}}$ may have not been transferred yet previously at time t_i. However, the update probabilities are chosen in a way to guarantee that no redundant update is triggered ($\forall s_j, s_i \in S, s_j = s_i : r(s_j, s_i) = 0$), i.e., the same context state is not retransmitted twice in a row.

The properties of the update policy with memory are too complex to be described with a single function. The reason is that we additionally need to express the fact that updating the same context repeatedly is a wasted operation. Therefore, we use the framework of a Constrained Markov Decision Process (CMDP) to model our problem. CMPDs are discrete time stochastic decision processes for describing systems, in which actions are taken on Markovian state transition systems. As an effect of taking an action, the system evolves from one state to another according to given transition probabilities. Moreover, changing a state leads to certain costs as well as rewards, which both depend on the particular action chosen. A solution of the CMDP is given by the probability of taking a specific action, such that the reward is maximized while costs

are limited by a configurable threshold.

In the following, we develop a representation of a CMDP that fits our problem description. For the purpose of our problem, reward corresponds to the gain in context accuracy and costs represent the energy consumption caused by the updates. The following are the components of our CMDP:

- **States**: The set of states known to the system is denoted as $X = (S \times S)$ and represents tuples of context states. The first element s_U of a tuple $(s_U, s_C) \in X$ refers to the last context transmitted to the consumer, while the second element s_C denotes the current context of a producer.

- **Actions**: The set of actions $A = \{update, no\_update\}$ reflects the update decisions we have to make. If action *update* is taken, an update message is sent to the consumer which contains the current context state of the producer, whereas no update is triggered when action *no_update* is chosen.

- **Transition Probabilities**: The transition probabilities in a CMDP are given as $p(i, a, j)$, defining the probability of moving from state $s_i \in X$ to state $s_j \in X$ when action $a \in A$ is taken. It holds that $\forall s_i \in X, a \in A : \sum_{s_j \in S} p(i, a, j) = 1$. We derive the CMDP transition probabilities from the Markov model that describes the evolution of the user's behaviour:

$$p((s_{U_1}, s_{C_1}), a, (s_{U_2}, s_{C_2})) =$$

$$\begin{cases} p(s_{C_1}, s_{C_2}), & \text{if } (a = update \wedge s_{U_2} = s_{C_1}) \\ & \vee (a = no\_update \wedge s_{U_2} = s_{U_1}) \\ 0, & \text{else} \end{cases}$$

  The transitions are defined in a way to reflect the result of the update decisions taken. Each time an context update is forwarded ($a = update$), the consumer acquires a new context state ($s_{U_2} = s_{C_1}$). Otherwise, if no update is taken ($a = no\_update$), the consumer still assumes the last received context state to be valid ($s_{U_2} = s_{U_1}$).

- **Costs**: The costs arise from the number of sent update messages and thus depend on the action $a \in A$ taken in the current system state $(s_U, s_C) \in X$:

$$c((s_U, s_C), a) =$$

$$\begin{cases} 0, & \text{if } (a = no\_update) \\ \frac{e_u}{\Delta_{t_s}}, & \text{else} \end{cases}$$

  Each time an update is taken we impose as cost the energy e_u required for a single update message, averaged over the sampling interval Δ_{t_s}. In contrast, sending no

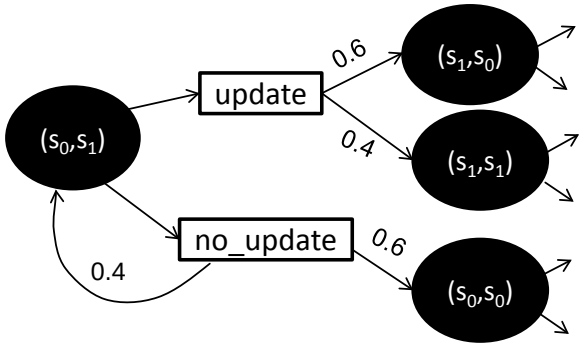


Figure 8.3.: Example of the CMDP for the update policy with memory

updates does not produce any cost.

- **Reward**: The reward measures the increase in accuracy depending on the action $a \in A$ applied to the current system state $(s_U, s_C) \in X$. The reward is defined as a function

$$ac((s_U, s_C), a) =$$

$$\begin{cases} w(s_C), & \text{if } ((a = update \wedge s_U \neq s_C) \\ & \vee (a = no\_update \wedge s_U = s_C)) \\ 0, & \text{else} \end{cases}$$

  The increase in accuracy is related to the relevance weight $w(s_C)$ of the current producer state s_C. Such an increase may happen in two cases. If the current context state equals the last transmitted state (i.e. $s_C = s_U$), the consumer context is accurate even if no update is sent. Further, the accuracy is increased if we send an explicit update after a context change has been observed (i.e. $s_U \neq s_C$). Note that we give no reward to unnecessary updates which would not benefit the accuracy at the consumer. This may occur when an update is triggered, although the consumer is already aware of this state (i.e. $s_U = s_C$). Since such an update would increase the cost without increasing the accuracy, it will be avoided by the optimization algorithm.

An example of a CMDP based on our definitions is shown in Figure 8.3. Given states $S = \{s_0, s_1\}$, the figure shows the evolution of the decision process starting from $(s_0, s_1) \in X$. This CMDP state encodes that the last update was sent for state s_0, and now a decision is required whether an update should be sent for state s_1. Depending on the decision made, a specific state transition will be triggered (associated with a defined probability). If the chosen action is *update*, the last updated context is now s_1. In contrast, if no update is made, the last transmitted context remains s_0 and only the current context evolves.

Based on the CMDP defined above, the goal is to find the probability for taking an

action (*update* or *no_update*) in each state of the decision process. It has been shown that the solution of the CMDP can be determined with LP [Alt99]. For this purpose, a translation of the CMDP into an LP problem is required as follows:

$$\text{minimize} \sum_{x \in X} \sum_{a \in A} \rho(x,a) \cdot ac(x,a) \tag{8.11}$$

$$\text{subject to} \sum_{x \in X} \sum_{a \in A} \rho(x,a)[\delta_y(x) - p(x,a,y)] = 0, \ \forall y \in X \tag{8.12}$$

$$\text{and} \sum_{x \in X} \sum_{a \in A} \rho(x,a) = 1 \tag{8.13}$$

$$\text{and } \rho(x,a) \geq 0 \ \forall x \in X, a \in A \tag{8.14}$$

$$\text{and} \sum_{x \in X} \sum_{a \in A} \rho(x,a) \cdot c(x,a) \leq E \tag{8.15}$$

In equation (8.11), we state our optimization function. The unknown variables are of the form $\rho(x,a)$, which indicate the joint probability distribution of taking action a in state x, i.e. the probability that such a combination exists in the decision process. As the resulting solution needs to be ergodic (i.e. all states are reachable), the constraint in (8.12) is required such that the outgoing and incoming rates are equal [Alt99]. Additionally, the constraints in (8.13) and (8.14) are necessary to define $\rho(x,a)$ as a consistent probability measure. Finally, the constraint in (8.15) refers to the expected energy cost, which results from the probabilities of taking action *update* in all possible CMDP states. These costs may not exceed the given energy budget E.

Based on the solution of the CMDP, the parameters of the update policy can be derived in a subsequent step. Formally, we are interested in the conditional probability for applying action *update* in each state $x = (s_U, s_C) \in X$. This probability can be computed using the following equation:

$$r(s_U, s_C) = \frac{\rho((s_U, s_C), update)}{\sum_{a \in A} \rho((s_U, s_C), a)} \tag{8.16}$$

According to Bayes' rule, we can derive $r(s_U, s_C)$ as the conditional probability from the joint probability returned by the solution of the CDMP. This way, we can define the policy r over all state pairs (s_U, s_C), where s_U represents the last updated state at a consumer and s_C denotes the current state at the producer.

8.3. Predictive Sensing and Update Protocol with Hard Energy Bounds

In this section, we extend our work to account for a another critical source of energy consumption. For the previous approach, we have assumed to have no explicit control

over the context sensing process. Therefore, context sensing has been performed as a continuous process running in the background with constant sensing interval Δt_s. In our extended problem statement, we relax this assumption and address the problem of controlling sensing and communication operations in an integrated way.

8.3.1. Extended Problem Statement

In order to recognize a context state at a specific time, a sensing operation is required on the device. We assume that every sensing operation consumes e_s amount of energy. Hence, sampling a new context with minimum sensing interval Δt_s yields the highest energy cost. In order to save energy, the mobile device may decide to selectively increase the sensing interval and introduce duty cycles, so that less sensing operations are required on average. Formally, let $T' \subseteq T$ denote the sensing schedule given by the times of when a sensing operation shall be performed. Given a particular schedule T', the instantaneous sensing cost $\omega(t)$ at a specific time $t \in T$ are then defined as:

$$\omega(t) = \begin{cases} e_s, & \text{if } t \in T' \\ 0, & \text{else} \end{cases} \tag{8.17}$$

Each time a new context state becomes available, the producer has to decide whether this information should be forwarded to a consumer. This introduces additional energy costs caused by update messages for transferring the context information over the wireless channel (see Section 8.2.1). Hence, since both sensing and communication operations are required to enable those updates, the total energy cost is given by

$$ec = \frac{\sum_{t \in T} \omega(t) + \mu(t)}{|T| \cdot \Delta t_s}$$

which sums the energy overhead (measured as J/h) required for sensing $\omega(t)$ and communicating $\mu(t)$ context information on a user's mobile device.

In terms of the context accuracy ac, further uncertainty is introduced through the selective sensing process. Due to the scheduled duty cycles, context sensing operations are suppressed so that time gaps occur where knowledge about possible context changes is missing. During these gaps, no fresh context information becomes available, so that consumers have to assume that the last seen context state is still valid. However, if the actual user state has changed in the meantime, this information is inaccurate and the true context state remains unnoticed. Hence, the selection of appropriate sampling intervals has a major impact on the resulting context accuracy. In particular, since the probability for state transitions may significantly differ among various user states, sensing intervals should be chosen which consider the underlying user behaviour characteristics to minimize the loss of accuracy. For measuring the context accuracy we refer to Equation 8.3, where we compare the context information that is available at

the consumer with the producer's actual context state over all time slots (see Section 8.2.1).

The fundamental problem addressed in this section is to control the sensing and communications operations on a mobile device to respect a given energy budget E. Formally, the goal is to determine the optimal set of samples at times T' and upload at times U such that the following constrained optimization problem is solved:

$$\text{maximize} \qquad \frac{\sum_{t \in T} \lambda_w(t)}{|T|} \tag{8.18}$$

$$\text{subject to} \qquad \frac{\sum_{t \in T} \mu(t) + \omega(t)}{|T| \cdot \Delta t_s} \leq E \tag{8.19}$$

The optimization goal (8.18) strives for the maximization of the context accuracy ac at the consumer. The optimization constraint (8.19) limits the total energy cost ec at the producer by E. Note that in contrast to the problem addressed in Section 8.2.1, we additionally need to decide for suitable sensor duty cycles in order to restrict the energy consumption on the device. Previously, only communication operations have been considered. In the next sections, we present a new sensing-and-update protocol that makes use of the knowledge about the prospective user behaviour to solve this optimization problem.

8.3.2. Approach Overview

We propose a novel predictive sensing-and-update protocol (PreSUP) to address the problem stated above. For this purpose, we apply a stochastic optimization framework which is based on a similar methodology as previously introduced in Figure 8.1. Key to our approach is a prediction-based approach to find an effective sensor sampling strategy which exploits the information from a Markov model to assign specific sensor duty cycles to different user states. The idea of this approach is that the sensor duty cycles should be conditioned on the specific characteristics of a user's behaviour (i.e. the probabilities with which a user transitions among different user states) in order to minimize the resulting loss of accuracy. At the same time, based on the selection of suitable duty cycles, we can control the costs of sensing and, implicitly, also the communication costs as explained in the following.

More precisely, given a particular context state, PreSUP preferably delays the next context sensing operation in situations where we expect the prospective user behaviour to persist, i.e., no changes to the current user state are expected over upcoming sensing intervals. In contrast, if the user is expected to leave his current state soon with a high probability, a low sampling interval will be chosen to capture forthcoming state transitions so as to preserve the context accuracy as much as possible. To implement this strategy, let $A = \{\Delta t_s, 2 \cdot \Delta t_s, ..., n \cdot \Delta t_s\}$ denote the set of all possible duty cycles

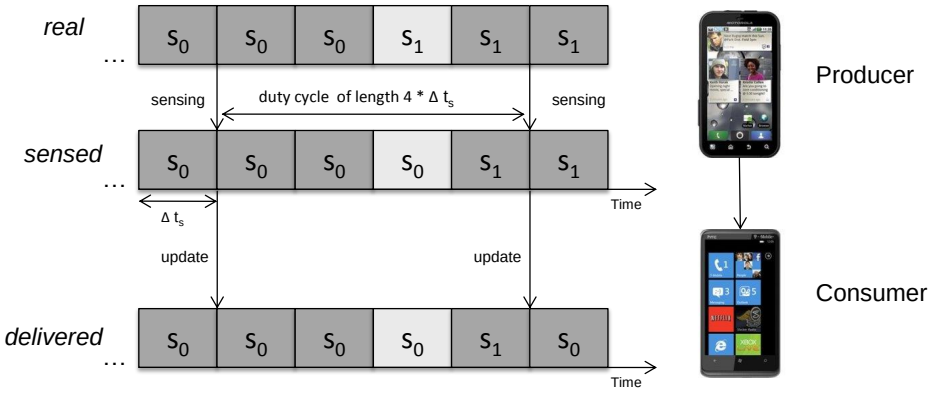


Figure 8.4.: Overview of our Predictive Sensing and Update Protocol (PreSUP)

among which we can choose. Then, PreSUP relies on a sensing policy formally defined as $s : (S \times A) \rightarrow [0, 1]$ to select duty cycles based on a probabilistic decision. More precisely, $s(s_C, a)$ indicates the probability of duty cycling the on-board sensors for a period $a = i \cdot \Delta t_s$ once state $s_C \in S$ has been observed. Note that $s(s_C, i)$ is associated with a specific state s_C, since distinct transition probabilities may be observed in different states which renders state-dependent schedules more effective. The best policy s is inferred through our optimization framework, which processes a user's Markov model to find the duty cycles that best matches a user's context patterns. While longer duty cycles are chosen in states where the Markov model suggests persistent behaviour, short duty cycles are employed when context changes are more likely. Hence, our approach is explicitly designed to automatically adapt to given user behaviour characteristics to find the optimal solution.

Consider the example in Figure 8.4, where a duty cycle of $4 \cdot \Delta t_s$ is applied on the producer to suspend the next sensing operation. The impact of the duty cycle on the context accuracy depends on the sequence of context states which have been missed. As over the next two sensing intervals the last sensed context state (s_0) remains valid, no accuracy is lost even though a new sensing operation has not been performed. Then, in the next interval, the producer's context state changes from s_0 to s_1. Due to the paused sensing process, this context change remains unnoticed until another interval elapses after which the duty cycle ends where s_1 is sensed and sent to the consumer. Hence, while just an inaccuracy for the length of Δt_s is introduced, the required energy only takes up for a fraction of $\frac{e_s+e_u}{4 \cdot e_s+e_u}$ over this period. For typical values of e_s and e_u (see our evaluation in Section 8.4.1), this corresponds to savings of 54%, which demonstrates the great potential for reducing the on-device energy costs when knowledge about a user's future behaviour is exploited for suspending sensing and communication operations.

Note that PreSUP does not consider separate sensing and update decisions. This means that context updates will be transmitted once a context change has been sensed. We

thus avoid useless sensing operations for acquiring context states which would then not be forwarded. Nevertheless, the communication cost which are consumed for every update message restrict the selection of possible sensor duty cycles. In particular, for every context change that has been sensed, a context update needs to be forwarded which consumes energy for its transmission over the wireless network. The selection of sensor duty cycles therefore has to consider both cost factors to satisfy the given energy budget. In the following, we present our approach of how to derive such a policy in an automatic manner.

8.3.3. Sensing and Update Algorithm

In this section, we first explain how PreSUP is applied on a mobile device to control the sensing and update process. Subsequently, we describe how to derive the sampling policy s using a stochastic optimization method which is based on a tailored CMDP model.

Algorithm 9 describes the on-device sensing and update process. First, the current state s_C is sampled for retrieving an up-to-date view on the producer's context (line 4). This causes the available energy budget to shrink by the amount of energy required for performing a single sensing operation (line 5). If a context change can be observed ($S_C \neq S_L$) or initially when the application has been launched ($S_L = null$), s_C is transmitted to the consumer. In either case, the energy budget is decreased by the amount of energy that is required for a single communication operation (lines 6-9). Then, a sensor duty cycle t_{sleep} is chosen from the sensing policy s (lines 10-18) by sampling from the distribution $s(s_C, i)$ associated with s_C. Afterwards, the sensor sampling process is suspended for the duty cycle of t_{sleep} to reduce the energy consumption (line 20). This whole process is repeated continuously: as soon as the duty cycle has elapsed, a new duty cycle is chosen which is specific to the sensed state s_C and any observed context change is transmitted over the network. To cater for inaccurate predictions which would result in energy overspending, we constantly monitor the energy consumption. If the energy budget is drained up to the point where only energy for one update message is remaining, we publish the context state with the highest stationary distribution (see Section 7.5) in a final update to maximize the consumer-side context accuracy for the rest of the application lifetime (line 22-23).

We model the properties of our sensing-and-update protocol as a CMDP. The CMDP allows us to reason over the effect a specific distribution of s would have on the resulting energy cost and context accuracy. To achieve this, we develop a tailored CMDP model with the following specification:

- **States**: The CMDP only deals with the known set of context states S. We do not further distinguish between producer and consumer states, since both share

Algorithm 9 Sensing and Update Protocol

```
1: s_L ← null
2: t_sleep ← null
3: while E · t_max ≥ 2 · e_u + e_s do
4:     s_C ← sample context
5:     E ← E − e_s
6:     if s_L = null ∨ s_C ≠ s_L then
7:         send update message(s_C)
8:         E ← E − e_u
9:     end if
10:    r ← rnd()
11:    for i = 1 to l do
12:        if r ≤ s(s_C, i) then
13:            t_sleep ← i · Δt_s
14:            break loop
15:        else
16:            r ← r − s(s_C, i)
17:        end if
18:    end for
19:    s_L ← s_C
20:    sleep(t_sleep)
21: end while
22: s_M ← s_r with r = arg max_{i∈{1,...,n}}{π_i}
23: send update message(s_M)
```

the same view on the evolution of the system according to our sensing-and-update protocol.

- **Actions**: The set of actions $A = \{\Delta t_s, 2 \cdot \Delta t_s, ..., n \cdot \Delta t_s\}$ comprises the different sensing duty cycles among which we can choose. The communication operation is not explicitly part of this set, as every recognized context change will be implicitly updated.

- **Transition Probabilities**: The transition probability $p(i, a, j)$ indicates the probability for moving from s_i to s_j after the chosen duty cycle $a = k \cdot \Delta t_s \in A$ has elapsed. $p(i, a, j)$ is based on the underlying transition probability of the Markov model and the chosen duty cycle. It can be derived as:

$$p(s_i, k \cdot \Delta t_s, s_j) = p_{ij}^k$$

where p_{ij}^k denotes the k-step transition probability (cf. Section 7.5) to estimate the probability for the user to be in s_j after the sensor has stayed idle for a time

of $k \cdot \Delta t_s$. Hence, $p(i, a, j)$ expresses the probability for a state transition when the sensing process has been suspended for the chosen duty cycle a.

- **Costs**: To define the costs related to the sensing and communication overhead, we define two utility cost functions. Both cost functions will be later integrated to account for the total energy costs.

  With respect to the sampling costs, we define the function

  $$dur(s_i, k \cdot \Delta t_s) = k \cdot \Delta t_s$$

  that gives us access to the scheduled duty cycle associated with an action. The sensing cost incurred by this action are inversely proportional to the length of the duty cycle. More precisely, since an amount e_s of energy is consumed per sensing operation and exactly one sensing operation is executed for each sensing interval, an average energy consumption $\frac{e_s}{k \cdot \Delta t_s}$ is incurred for the entire duty cycle. As can be seen, the longer the duty cycle, the more energy can be saved.

  In addition, update messages are sent to inform remote consumers about the user's context. However, updates are only triggered for actual state changes. Therefore, the resulting average communication costs depend on the expected state change probability and can be specified as

  $$c_{update}(s_i, k \cdot \Delta t_s) = (1 - p_{ii}^k) \cdot e_u$$

  Given a state s_i, the probability for a state change can be derived from its self-transition probability. More precisely, the probability to be in s_i after an idle time of $k \cdot \Delta t_s$ (involving k time slots) is given by p_{ii}^k. This probability p_{ii}^k needs to be subtracted from 1 to derive the probability for an actual state change to any s_j with $j \neq i$. Each time such a state change occurs, it is reported to the consumer via an update message, requiring an amount e_u of energy for its transmission.

- **Reward**: The reward indicates the expected gain in context accuracy, which depends on the chosen duty cycle $a \in A$ in state $s_i \in S$. This accuracy gain is defined as

  $$ac(s_i, k \cdot \Delta t_s) = w(s_i) \cdot (1 + \sum_{t=1}^{k-1} p_{ii}^t)$$

  As discussed before, the context accuracy depends on whether a sensed context state s_i remains valid over the duration of a duty cycle. We therefore compute for each missed context sample the probability of being in state s_i. For this purpose, the k-step self-transition probabilities p_{ii}^k can be used. We add a probability of 1 to this sum, since sensing state s_i initially at the start of a cycle guarantees true knowledge. Also, we factor in the state relevance weight $w(s_i)$ to consider the application's interest in not missing the specific state s_i.

The CMDP components describe the basic characteristics of our sensing-and-update protocol, modelled as a stochastic control process. The solution of the CMDP problem yields the probability of selecting a sensing interval $a \in A$ in a given state $s_i \in S$. For this purpose, the CMDP needs to be translated to a representation which can be understood by a suitable problem solver. This representation is satisfied by the following system of functions and equations [Alt99]:

$$\text{maximize} \frac{\sum_{x \in X} \sum_{a \in A} \rho(x,a) \cdot ac(x,a)}{\sum_{x \in X} \sum_{a \in A} \rho(x,a) \cdot c_{sense}(x,a)} \tag{8.20}$$

$$\text{subject to} \sum_{x \in X} \sum_{a \in A} \rho(x,a)[\delta_y(x) - p(x,a,y)] = 0, \ \forall y \in X \tag{8.21}$$

$$\text{and} \sum_{x \in X} \sum_{a \in A} \rho(x,a) = 1 \tag{8.22}$$

$$\text{and } \rho(x,a) \geq 0 \ \forall x \in X, a \in A \tag{8.23}$$

$$\text{and} \frac{e_s + \sum_{x \in X} \sum_{a \in A} \rho(x,a) \cdot c_{update}(x,a)}{\sum_{x \in X} \sum_{a \in A} \rho(x,a) \cdot dur(x,a)} \leq E \tag{8.24}$$

In contrast to the previous LP formulation in Section 8.2.3.2, the underlying optimization function (8.20) for our problem is non-linear. The reason is that the consumer-side accuracy to be maximized is measured as a function per time. However, as the elapsed time between two sampling operations depends on the chosen duty cycles, the expected sensing interval needs to be considered in the denominator of the function. This results in a non-linear optimization function. We therefore apply Non-Linear Programming (NLP) to find the optimal solution. The constraints in (8.21), (8.22), (8.23) ensure that the resulting solution defines a valid probability measure. The constraint in (8.24) estimates the sampling and energy cost that result from applying the policy which may not exceed the given energy budget E.

Solving the NLP problem yields the probabilities $\rho(x,a), \forall x \in S, a \in A$, which define the joint probability that a state x occurs and the corresponding action a is executed. However, as expressed by our policy s, the probability of choosing a specific update interval is required. This can be derived as the conditional probability

$$s(s_C, i) = \frac{\rho(s_C, i \cdot \Delta_{t_s})}{\sum_{a \in A} \rho(s_C, a)} \tag{8.25}$$

from the joint probabilities by means of marginalization. We thus obtain the optimal assignment of probabilities $s(s_C, i)$ for choosing a duty cycle of length $i \cdot \Delta t_s$ in state s_C. Note that the NLP solving has to be computed only once and can be offloaded to a computer with more computational power and unlimited energy supply, if required. On the mobile device, only the resulting update policy is needed to drive the sensing-and-update process as explained previously.

8.4. Evaluation

In this section, we present a comprehensive analysis of the protocols developed in this chapter. For this analysis, we evaluate both classes of our protocols, the update-only protocol and the sensing-and-update protocol. In the following, we first we introduce our underlying evaluation methodology and then discuss the results gained from our evaluation study.

8.4.1. Evaluation Methodology

For the evaluation of the update protocols proposed in this chapter, we followed a similar setup as described in Section 7.7. Analogous to our previous evaluation method, we used real-world traces of human activities from the CenceMe project [MLF+08] to validate our protocols against realistic sequences of context changes. However, in contrast to our previous evaluation strategy, hard bounds on the amount of available energy E were given. Given a specific energy budget E, we analysed the protocol in terms of their ability to maximize the context accuracy. Context accuracy is measured as defined in Equation 8.3, quantifying the correctness of the context information perceived by a consumer. As we assumed all context states to be equally important, we defined a uniform relevance weight for the states, i.e., $\forall s_i \in S : w(s_i) = 1$. Analogous to the approach followed in Section 7.7, we performed our evaluation based on two different scenarios. One the one hand, we used a Monte Carlo approach to generate synthetic activity traces based on a Markov model that we learnt from the real traces. On the other hand, we performed the measurements directly against the actual sequence of context states from the real traces. By considering these two different scenarios, we analysed the vulnerability of our approach to violations of the Markov property which is critical for the exactness of our prediction model (see Section 7.7).

To quantify the energy usage of our protocols, we rely on assumptions about the energy overhead of sensing and communication operations. In terms of the sensing costs, we leverage on the profiled energy usage of the CenceMe mobile application [MLF+08]. According to this study, $e_s = 1.28$J is consumed on average for a sensing operation that classifies a user's activity based on collected acceleration samples. As the CenceMe trace records activities in constant periods of $\Delta t_s = 8s$, this time defines the minimum sensing interval which can be adapted by our algorithms. In terms of wireless communication costs, according to the empirical energy model prosed in [BBV09], $e_u = 3.2$J is consumed to transmit a context update in an isolated update message over GPRS networks (cf. Section 6.3). Since the encoding of context states requires little overhead, our update messages are of minor size so that most of the consumed energy results from the GPRS tail energy. In the following, we give insights into our evaluation results for the protocols developed in this chapter - first, we focus on the update-only protocols which merely consider network transmissions, and then we analyse our sensing-and-update protocol

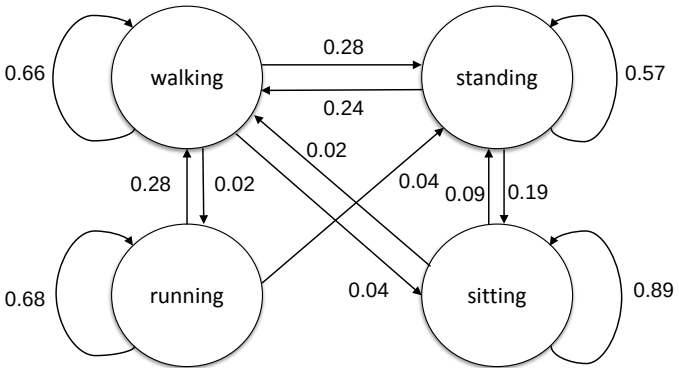
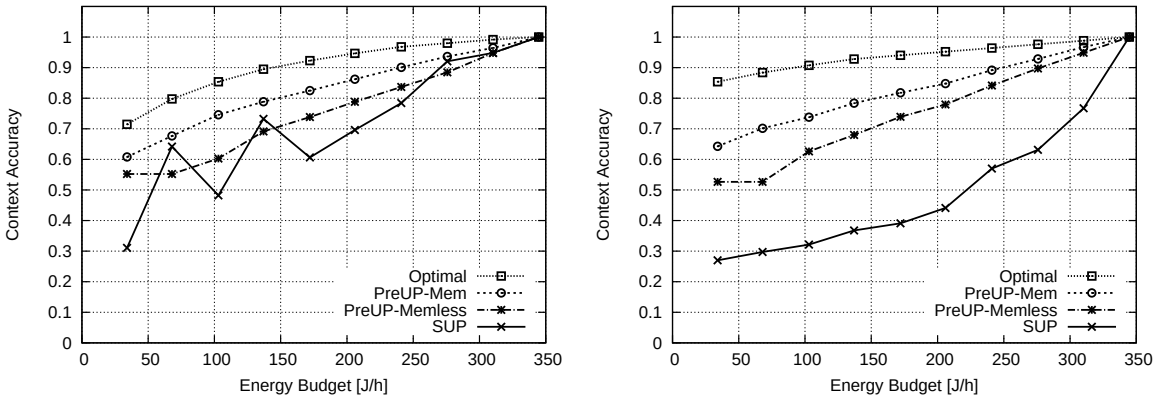


Figure 8.5.: Learnt Markov model of user activities based on the CenceMe trace

that additionally addresses the problem of sensor scheduling. Since our protocols are based on stochastic update decisions, we repeatedly execute the experiments and plot the context accuracy averaged over 100 runs for the same energy budget.

8.4.2. Stochastic User Model

For the explanation of our evaluation results, an analysis of the degree of user behaviour dynamics is important. Figure 8.5 depicts the Markov model that was learnt from the context transitions found in the CenceMe trace. For the sake of clarity, only transitions with a significant probability are included in the figure. The self-transitions serve as a good indicator for the degree of user behaviour dynamics in different states. The state 'sitting' exhibits the highest self-transition probability, i.e., the user tends to remain in this state for a number of subsequent samples with high probability. Consequently, this behaviour offers great potential for saving sensing and update cost, since, having sensed this state, the probability to assume a correct state in the following time slots is high even if no new sample is obtained. The states 'running' and 'walking' have lower self-transition probabilities, since these activities are performed for a shorter duration by the user. The lowest self-transition probability is associated with the state 'sitting'. In this state, the chance to miss any state transition during subsequent samples is more significant and may affect the context accuracy much stronger. This discussion reveals that the distribution of transition probabilities has a great impact on the performance of our protocols. Therefore, no general solution can be proposed for arbitrary application scenarios, since each scenario may have different user behaviour characteristics. Therefore, our update protocols are aligned to the specific behaviour of the user as exposed by the Markov model. While we have used the self-transition probabilities in the discussion above as a heuristic to explain feasible optimizations, an analysis over all transition probabilities is performed by the CMDP to find the optimal solution.



(a) Synthetic traces (b) Real traces

Figure 8.6.: Context accuracy of update protocols with hard energy bounds

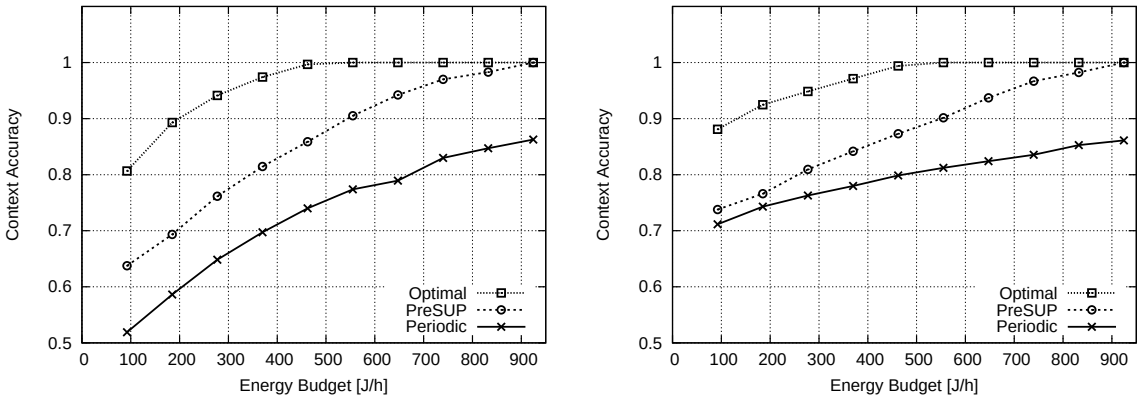
8.4.3. Update Protocols

In the following, we discuss the experimental results for our predictive update protocols (PreUP-Memless and PreUP-Mem) across various energy budgets. In our evaluation, we compared our protocols with two alternative approaches. As one alternative, we used a deterministic state-triggered update protocol (SUP). SUP transmits each occurring state change until the allocated energy budget is completely drained. However, no further knowledge about user behaviour is exploited to influence the update decision. Therefore, this protocol serves as a baseline to study the benefits of our prediction-based protocols. Additionally, we have determined the optimal solution (i.e. highest achievable accuracy for a given energy budget) using a Dynamic Programming (DP) algorithm that selects the best updates. In contrast to the online approaches we have developed, this approach is based on a complete view on the entire sequence of user behaviour which can only be performed offline. As the offline algorithm requires more knowledge than is available at runtime, it is used as a reference only to study the theoretically best possible solution.

The results for applying our update protocols to the synthetic context trace are shown in Figure 8.6 a). As expected, the accuracy generally increases with higher energy budgets. However, SUP shows alternating increases and decreases in context accuracy for rising energy budgets. This is counter-intuitive, since a higher energy budget allows for more updates to be sent, which should increase the context accuracy. This observation can be explained by the fact that, in case of a drained energy budget, the last update made has a major effect on the context accuracy for the remaining run-time of the application. Since SUP transfers all context changes as soon as they are observed, it tends to quickly drain the given energy budget. As a result, since the last updated context state may not frequently occur afterwards and no further update can be sent, the context accuracy at the consumer may suffer from this approach. Our predictive update protocols (both PreUP-Memless and PreUP-Mem) circumvent this effect, since

the frequency of updates is adapted to the amount of available energy. Instead of simply transmitting every context change as soon as it occurs, the more effective updates are preferred over the less relevant ones based on the analysis of the user's future behaviour. Also, the last update is chosen to be the state with the highest stationary distribution, thus maximizing the context accuracy for the remaining run-time. As a result, PreUP-Memless improves SUP by 12% on average across all energy budgets, and PreUP-Mem outperforms it even by 24%. Compared to PreUP-Memless, PreUP-Mem achieves a higher context accuracy at a consumer, since redundant updates are avoided so that the context accuracy increases for every update message (cf. Section 8.2.3.2). The theoretically optimal solution is on average 10% better compared to PreUP-Mem. The reason is that the offline approach can take benefit from perfect knowledge about the actual sequence of a user's context states, so that no prediction error can occur that would cause suboptimal update decisions. This is especially beneficial for lower energy budgets where only few update messages may be sent and accurate predictions are required to identify effective updates. However, even though the global knowledge required to determine the optimal result is not available at run-time, PreUP-Mem achieves a good approximation of the optimum. This demonstrates that the predictions are reliable most of the time, so that effective updates are chosen.

Figure 8.6 b) illustrates the results for the real context trace. As can be seen, our protocols are able to achieve even better accuracy improvements over the different energy budgets in this case. While the performance of our update protocols are similar to the experiments with the synthetic trace, the results for SUP are much worse. This stems from the fact that a lot of context changes can be observed in the real trace that only last for a short duration. Since SUP transmits these context changes without further analysis of their effectiveness, the given energy budget is quickly exhausted and the transferred context updates become invalid soon. On average, PreUP-Memless improves the context accuracy by 63% compared to SUP, and PreUP-Mem even by considerable 84%. The gain is particularly high for lower energy budgets, where only few update messages can be afforded that need to be carefully chosen to guarantee accurate context information at the consumer. In this case, PreUP-Memless and PreUP-Mem select context changes which are much more useful to maximize the context accuracy. As expected, PreUP-Mem shows a better performance than PreUP-Memless through making more accurate predictions, resulting in more effective updates. It can be seen that a substantial amount of energy is required to reach a perfect accuracy at the consumer, which can only be achieved when every context change is transferred. Nevertheless, our update protocols are able to achieve a high context accuracy already for lower energy bounds. For instance, using PreUP-Mem we can achieve 85% accuracy for energy budgets that are assigned 60% of the maximum energy required to deliver perfect context accuracy. In contrast, 95% of the energy are required by the SUP protocol to reach the same 85% accuracy level. The optimal solution computed by our DP offline algorithm is on average 14% better in context accuracy compared to PreUP-Mem. Considering the fact that the optimal solution serves only as a theoretical



(a) Synthetic traces (b) Real traces

Figure 8.7.: Context accuracy of sensing-and-update protocol with hard energy bounds

reference, we can conclude that PreUP-Mem achieves close-to-optimal performance. As a consequence, our evaluations demonstrate that we can gain significant improvements when adapting the update protocols to the predicted behaviour of the user. This even holds for the real trace, in which predictions are much more susceptible to deviations from the user's actual behaviour.

8.4.4. Sensing and Update Protocol

We compare our predictive sensing-and-update protocol (PreSUP) with a periodic time-based protocol (Periodic), which performs a sensing operation in constant time intervals and reports the sampled context to the consumer. The time interval Δt_p is chosen as the minimum period which can afford both a sensing and communication operation to be executed for a given energy budget E, i.e., $\Delta t_p = \frac{E}{e_s+e_u}$. We also include the theoretical optimum as a reference, that is calculated by an offline algorithm using a Dynamic Programming (DP) approach which determines the best duty cycles for a sequence of context states considering both the sensing and communication cost. Note that we cannot compare PreSUP with the update protocols evaluated in the previous section (PreUP-Memless and PreUP-Mem), which have been designed to optimise the wireless communication overhead, but not the sensing process. Since these protocols require a new context state to be continuously sampled, full sensing cost arise so that we cannot meet the limitations on the total energy consumption for all energy budgets E subject to this evaluation.

First, we look at the results for the synthetic trace shown in Figure 8.7 a). As can be seen, PreSUP increases the context accuracy at the consumer by 18% on average compared to Periodic. The increase is almost constant over all energy budgets. This demonstrates that PreSUP is able to exploit a constant fraction of the available energy much more

effectively. The reason is that updates executed in constant periods do not account for the fact that the different states exhibit heterogeneous transition probabilities. This makes particular states with more persistent behaviour much more suitable for longer duty cycles. In contrast, PreSUP is based upon this rationale, assigning longer duty cycles to states in which the user remains, while shorter duty cycles are preferred in states where state changes are much more likely. For our given scenario, especially the state 'sitting' is identified to be the most persistent one where longer duty cycles can be applied to save energy and retain a high context accuracy level despite of missed context samples. The theoretically best possible accuracy computed by the DP algorithm is on average 13% better compared to PreSUP. This is because global knowledge of the entire sequence helps identifying the very best sensing and update intervals. In contrast, PreSUP must rely on predictions that may cause sub-optimal sensing intervals to be chosen. For instance, in case the user stays spends more time in a given state than predicted, a longer duty cycle would have been more effective for saving energy and preserving the accuracy. Being close-to-optimal, though, the results suggest that our approach can effectively compensate the lack of global knowledge that is exploited by the offline approach.

Figure 8.7 b) illustrates the results we gained for the real-world trace, where the same ranking can be observed among the different protocols in terms of their performance. Specifically, PreSUP is superior to Periodic by achieving a relative improvement in context accuracy of 10% over all energy budgets on average. We can notice that, in contrast to the synthetic trace, the Periodic approach shows a better performance since long sequences of context states occur in the real trace where the user constantly remains in the same state. As a consequence, under the Periodic scheme often a context state is sensed with a significant occurrence probability that re-occurs over the period of a duty cycle. Note that in statistical terms the divergence of the real trace from the synthetic trace is caused by violations of the Markov property, i.e., the fact that the distribution of the state dwell times does not strictly follow a geometric distribution. Nonetheless, even though this Markov assumption does not strictly hold, PreSUP is able to find more effective duty cycles when adapting the sensing intervals to the state transition dynamics as encoded in the Markov model. In particular, PreSUP significantly improves the context accuracy compared to Periodic for increasing energy budgets. Even with more available energy, Periodic often senses the same state again with each consecutive sample. In contrast, PreSUP selectively allocates the available energy to recognize state changes, and therefore takes better advantage from more available energy. Since PreSUP can achieve this despite the fact that the dwell times do not strictly obey the Markov assumption, we can conclude that our approach is very robust in the presence of uncertainties and variances in the real-world data. On average, the theoretical optimum is 10% better compared to PreSUP. This performance gain is only possible using global knowledge about the entire sequence of all context occurrences. As the evaluation shows, PreSUP is able to infer this knowledge from the user's typical behaviour so that a high level of context accuracy can be achieved.

8.5. Related Work

The design of efficient approaches for acquiring and distributing context data has received a lot of attention in the field of mobile computing. Two major directions have been pursued in the recent literature, which will be discussed in the following: update protocols to communicate context data over wireless networks as well as approaches for energy-efficient sensing on mobile devices.

In terms of update protocols, the idea of considering a strict upper limit on the energy consumption is a novel research problem which has not been explicitly addressed in prior research. Instead of maximum energy consumption, earlier update protocols have been tailored towards other optimization criteria, which are relevant in their context of usage. For efficiently tracking users in cellular wireless networks, update protocols have been proposed which are able to achieve a good trade-off between update and paging costs [BD99, RMD07]. Noy et al. have analysed time-based, movement-based and distance-based protocols in terms of their trade-off characteristics [BNKS94]. However, the optimization of the update process has been primarily addressed from the perspective of a mobile network provider. For reasons of scalability, mobile network providers are interested in a minimum load on the cellular network. As mobile devices are only seen as the periphery of the network in these scenarios, their energy consumption has not been explicitly investigated.

For tracking the geographic positions of mobile objects with integrated GPS receivers, update protocols have been developed to transfer position data from mobile devices to remote location servers [WSCY99]. The updates adhere to spatial inaccuracy bounds, guaranteeing that the positions stored by the location servers deviate from the true user positions by a maximum distance only. Leonhardi et al. have adopted time-based and distance-based update protocols as known from cellular networks to the specific properties of moving objects whose geographic positions shall be tracked [LR01]. Since updates messages are sent whenever the inaccuracy conditions is about to be violated, a strict limit on the resulting energy consumption is not the primary goal of these protocols. Moreover, in our work, we focus on generic discrete context such as user activity or symbolic locations, where spatial distance metrics and assumptions about the movement of mobile users such as their maximum speed do not hold.

As an extension of the location-based update protocols, recently update protocols have emerged for the distribution of discrete context data [MPF+10]. These protocols allow for reducing the number of update messages by tolerating periods where interested consumers have inaccurate context information. However, it remains unclear how to configure these protocols to restrict the energy consumption to a given limit. Consequently, these protocols only work in a best-effort manner and may consume an arbitrary amount of energy. In contrast, we have shown how to derive a protocol configuration for transferring streams of discrete context data under hard energy bounds. The protocol we have proposed is not only designed to keep a given energy limit, but

also to maximize the quality of the context data reported to consumers.

Besides updates to remote receivers of context data, also the sensing process on a mobile device has attracted tremendous research interests [MLF+08, LPL+09, RMM+10]. As continuously sampling sensors heavily drains the batteries of mobile devices, a more selective control of the sensing process has been advocated. Key to save energy on the mobile device are approaches to suppress redundant or irrelevant sensor readings. More precisely, it has been shown that a substantial amount of energy can be saved when turning the sensors off opportunistically, e.g., while the users dwells at the same location [KLGT09], or during movements away from points-of-interest [BDR12b, WDR10]. However, these algorithms have been designed as best-effort approaches where the goal is to reduce the sensing costs without creating a loss in context accuracy for applications. To achieve this, opportunistic optimization strategies are used, which can only be applied in special occasions only depending on the user's behaviour. As it is not feasible to anticipate the occurrences of these occasions a priori, hard guarantees with respect to the consumed energy cannot be provided.

In contrast, energy bounds on the sampling process are considered by Constandache et al. [CGS+09], who propose a sensing scheme that uses only a limited number of position readings to estimate the locations of mobile users, considering the characteristics of different positioning technologies. For this purpose, the sensing operations are distributed over points in times at which the user is expected to change his location, thus avoiding redundant position readings to decrease the localization error as much as possible. A similar proposal has been made by Wang et al. [WKZA10] for a discrete model of context data to sense the user's activity. A Markov-optimal sensing plan is derived to schedule sensor readings at times so as to minimize the loss of accuracy. However, in contrast to our work, these approaches neglect the aspect of communication completely. As mobile sensing applications are meant to forward sampled context data over wireless data channels, the overhead of communicating context updates is an integral part of the total energy costs on the device. Therefore, an integrated approach is required which incorporates both cost factors into the optimization of the update process on a mobile device.

Based on this analysis, we conclude that we propose the first comprehensive approach for optimising update protocols suitable for discrete context data with respect to the total energy costs resulting from sensing and communication. This enables applications to be executed in a way that satisfies the energy constraints of mobile devices much more accurately, while maximizing the quality of the context delivery.

8.6. Summary

In this chapter, we have proposed novel update protocols for dealing with state-based context data on energy-constrained mobile devices. We have addressed the problem of

maximizing the accuracy of context perceived by a consumer under a given energy budget and presented update protocols that are able to exploit predictions of a user's behaviour to give priority to the most valuable context updates. For this purpose, we have modelled the distributed update process between context producer and consumer as a stochastic decision problem, and proposed algorithms that exploit context predictions to implement effective update strategies. We have studied this problem in two variants: First, we proposed an update protocol to identify the optimal sequence of update messages to inform consumers about context changes, and then we presented a sensing-and-update protocol which is also able to control the duty cycles of the on-device sensors. For both problems we have developed a representation as a Constrained Markov Decision Problem, which allowed us to leverage on Linear Programming (LP)/Non-Linear Programming (NLP) to solve the optimization problem and find the best update decisions which match given user behaviour characteristics.

In our evaluation, we have demonstrated for a real-world context trace from a mobile social networking application that our predictive protocols are much more effective than protocols that ignore knowledge of future user behaviour. Hence, our work contributes important techniques to support the growing field of mobile sensing applications with energy-aware design principles which are required to gain more practical relevance and achieve high acceptance among end users.

Part IV.

Summary and Outlook

Chapter 9

Summary

Deep knowledge about a user's context is key to the idea of context-aware computing. In the past, mainly the current context of users has been accessible to context-aware systems, e.g., the user's location at present. To increase the degree of proactiveness in these systems, not only awareness of the current context, but also of the user's future context is vital to develop novel predictive applications, e.g., mobile guide systems able to generate recommendations according to the user's next location visits. However, as this information is encoded in the daily routines of mobile users and not directly accessible to applications, context prediction is a non-trivial task which requires sophisticated methodologies to discover patterns of context changes from observation of the user's behaviour. Developing context prediction techniques that are able to support the idea of proactive computing in an effective manner is an open research problem which is not sufficiently addressed by previous approaches.

In this thesis, we have addressed open questions in the field of context prediction along two major lines of research. First, we have proposed new context prediction models for increasing the expressiveness and accuracy of predictions compared to existing prediction systems. The focus of this research has been on comprehensive stochastic models that allow for representing patterns of context changes in the user's behaviour in a machine-processable way. Based on these models, we have developed algorithms for performing stochastic inference over these models to compute reliable and expressive predictions. Second, we have shown that context prediction methods can be exploited to increase the efficiency of mobile sensing applications, a new class of mobile systems in the focus of current research. Since these applications are affected by high costs for sensing and communicating context data, we have presented algorithms to optimize their operation in terms of the trade-off between energy consumption and the accuracy of context information which is reported to interested consumers. In the remainder of this chapter, we present a summary of the contributions of this thesis.

9.1. Context Prediction Models

In the first part of this thesis, we have addressed the challenge of improving context prediction systems in terms of both prediction accuracy and prediction expressiveness. Prediction accuracy is one of the most crucial performance criteria of context prediction systems. Therefore, developing methods for increasing the accuracy of predictions is a major goal addressed by researchers. Previous context prediction system do not consider additional available domain knowledge that can be used to improve the predictions. In these systems, only single-dimensional context is assumed, while the interdependencies among different context types are neglected as further source of information. However, such interdependencies can make predictions much more accurate, since the occurrence of future context states may be conditionally dependent on this information.

To address this issue, we have proposed a context prediction scheme that is able to exploit domain knowledge from pervasive flow systems, which reveal the activities performed by humans in process-oriented applications, e.g., health-care scenarios. For this scheme, we have developed a new context predictor, which is able to encode the evolution of the user's context in relation to his executed flow activities, and algorithms to infer future context trajectories in a transition system of two-dimensional states. In summary, our core contributions are:

- We have proposed a new context predictor, the flow predictor, which is able to reflect the conditional dependency of context changes on flow activities as well as the past context history. The flow predictor is an extension of a classical Markov model with the flow activity as an additional random variable.
- A learning algorithm has been presented that automatically constructs a flow predictor from a context history and associated activity information. The algorithm determines the transition probabilities among two-dimensional context states to encode the probability of an activity-dependent context change.
- We have developed a prediction algorithm to determine the series of future context states that is most likely to occur. Since high computational overhead is involved in exploring the transition system, an efficient inference strategy has been proposed to reduce the search space.
- In our evaluation, we have shown that our context prediction scheme can exploit dependencies on activity information to significantly improve the accuracy in predicting future context changes. In case this dependency is not fully given, our predictor does not perform worse than a classical prediction approach.

In addition to high prediction accuracy, also rich prediction expressiveness is a desired feature to support a wide range of proactive applications. Prior prediction systems only determine the most probable next context as prediction and therefore suffer from a rudimentary degree of prediction power. In particular, any relation to real-time (i.e.

the time period within which a future context should occur) and further, more flexible prediction semantics are not supported. This severely restricts the kind of predictions which applications can leverage on.

We have proposed a new context prediction approach using methods of stochastic model checking to allow for time-dependent, semantic queries. While model checking is known as an effective means of verifying formal system, we have adopted and extended these techniques to render them usable for the prediction of human context. As a result, we have presented the first approach that combines probabilistic reasoning and logic-based querying semantics to allow for more expressive context predictions. Our contributions can be summarized as follows:

- We have proposed a machine-processable model of the user's real-world temporal behaviour. The model is based on a Semi-Markov Model and has an explicit representation of the dwell time behaviour in different user states.
- Temporal logics has been proposed as a formal query language for formulating context predictions. The query language provides various temporal operators with well-defined semantics for achieving a high degree of expressiveness.
- Novel context prediction algorithms have been developed based on stochastic model checking techniques. For this purpose, existing model checking algorithms have been extended with run-time semantics required to predict human context with dynamically evolving state dwell times
- In our evaluation, we have validated our approach for a real-world scenario from the domain of health-care using metrics from information retrieval. The results have shown that we can achieve a high trade-off between precision and recall for a suitable threshold configuration of the probability which determines when predictions should be delivered to applications.

9.2. Context Prediction in Mobile Systems

In the second part of this thesis, the benefits of context prediction in conjunction with mobile systems has been explored. To this end, we have studied mobile sensing applications, a class of mobile systems that has recently emerged from the wide availability of sensor-enabled mobile phones. In these applications, the sensor capabilities of mobile phones are exploited to constantly distribute information about the dynamic context of users to interested parties over cellular networks. Since these applications are affected by a high overhead of sensing and communication operations, we have concluded that the incurred energy consumption is critical for the limited battery capacities of mobile devices which can negatively impact the user's experience. Therefore, we have investigated new strategies for optimising mobile sensing applications in terms of the energy consumption required for delivering context updates to interested consumers.

To address this challenge, we have proposed new prediction-based update protocols for energy-efficient context updates among producers and consumers of context data. The protocols are based on the idea that, by predicting context changes locally at the consumer, the energy consumption on a mobile can be significantly reduced, avoiding costly transmissions over the cellular network to forward context updates. Our contributions can be summarized as follows:

- We have proposed a prediction-based update mechanisms that relies upon a shared prediction model that is known both to the producer and consumer of mobile user context. The prediction model is based on a discrete Markov model, which can be learnt from the user's past context changes.
- Four different prediction algorithms have been developed to exploit a Markov model for predicting missing context updates. Given a future time, each prediction algorithm estimates the context state in which the user is expected to be at that time based on the last known update.
- Two basic update protocols, a time-based and a deviation-based protocol, have been proposed to manage the energy/accuracy trade-off on the mobile device. With these update protocols, the degree to which the predicted context at the consumer deviates from the actual context of the producer can be effectively controlled.
- Our evaluation has shown that the prediction-based update mechanism can achieve improved trade-off characteristics in case the real-world behaviour adheres to the Markov property, whereas violations of this property cause inefficiencies due the inherent restrictions of Markov model statistics. The lessons learned from the evaluation study has inspired a more robust prediction-based update strategy, which we then introduced in the following chapter.

Drawing on the results from this research, we introduced an improved approach that is able to provide hard guarantees about the energy consumption on mobile devices. To this end, we have developed novel predictive update protocols for mobile sensing application, addressing the following optimization problem: given a limited energy budget for update operations, an optimal update schedule has to be found that maximizes the context accuracy at a consumer. To solve this problem, we have developed a model of a stochastic decision process which exploits knowledge about the future behaviour of mobile users to anticipate the effectiveness of various update decisions and infer the optimal protocol configuration. In more detail, we have made the following contributions:

- Two variants of a constrained optimization problem have been formalized to describe the tasks of informing a remote party about a user's context changes under a given energy budget. The variants differ in their assumption about whether the on-device sensors can be duty-cycled or not.
- A mathematical model of the update process between producer and consumer has been proposed. The model is based on a Constrained Markov Decision Process

(CDMP) and describes the energy costs and accuracy gains with respect to the different update decisions which can be made.

- In order to solve the optimization problem, a translation of the CMDP into a Linear Programming (LP)/ Non-Linear Programming (NLP) problem has been described. The solution reveals a probabilistic update schedule to control the context update process on a mobile device and achieves maximal context accuracy within the given hard energy bounds.
- In our evaluation, we have demonstrated for a real-world context trace from a mobile social networking application that our predictive protocols are far more effective than protocols that ignore knowledge of future user behaviour to deal with limited energy budgets.

In conclusion, the concepts and results of this thesis represent major contributions to the field of context prediction for the development of more intelligent proactive mobile systems. The context prediction models and prediction algorithms we have developed fill the gap of existing approaches in forecasting state-based context data with high accuracy and expressiveness. This will strongly help to advance the capability of context-aware systems to become much more intelligent in responding to the future needs of mobile users. Moreover, we have demonstrated that predictions about a user's future behaviour can be also fed back to mobile systems to vastly improve their energy efficiency in sensing and reporting a user's context changes. Since our algorithms are applicable to any forms of discrete context such as location or activity information, we can thus guarantee a wide exploitation of our contributions in current and future generations of context-aware mobile systems.

Chapter 10

Outlook

In this final chapter, we discuss directions for future research in the area of context prediction. Interesting opportunities for research activities are discussed which build on the results of our work, but have been beyond the scope of this thesis.

Life-long learning. In current research, only context data sets of limited temporal scope are available. The opportunities to study user behaviour over longer time periods are thus severely limited. In the future, the scale of available context data sets will be further growing with widely deployed context recognition technologies. As a result, the activities of humans are soon recorded over extensive periods of time, spanning not only a couple of days, but several months and most probably entire years. With respect to such extended time scales, humans usually go through different stages in their life which might involve significant behavioural changes. For instance, as soon as a student graduates from university, he will change his routine behaviour, possibly move to another city to find a job and travel on new ways during his everyday life.

These changed habits and routines must be taken into account for providing accurate context predictions. Therefore, it is vital to recognize behavioural drifts in a user's context and design learning algorithms which are able to discover and remove outdated data. Different techniques have been used in computer science to deal with the ageing phenomena of data, i.e., data where the past behaviour should be incorporated to a lesser degree than fresh data. For instance, the exponential moving average is a popular technique for smoothing historical data which exhibits changes over time. In future work, it should be explored how suitable these tools are for implementing life-long learning techniques in the field of context prediction. Based on the availability of large-scale data sets, realistic evaluation studies will be possible to compare the performance of different learning algorithms that may be used to solve this problem.

Large-scale context prediction. With our context prediction system, we have explored new ways of forecasting the behaviour of single persons. As a possible extension to this approach, the behaviour of groups could be predicted. For instance, proactive applications could be interested in knowing which group of people might execute a certain activity at a particular place in the future time. This prediction could be used for various purposes, e.g., for delivering traffic jam warnings to all car drivers for which the prediction holds. Even though such a prediction is in principle feasible with our approach, this raises fundamental challenges in terms of the scalability of the system, since not only a single user's prediction model, but those of all registered users would have to be evaluated.

Therefore, a system architecture is required that scales with the number of users for answering queries that involve entire groups of users. To accomplish this, new indexing techniques are to be developed that help restricting the number of evaluations that have to be made. The novelty of these methods would be that the index is not based on past or current data, but on assumptions about the users' future behaviour. For instance, in case of location prediction, the index could encode the future locations that can be potentially reached by certain users. Given such an index, those users who satisfy a prediction could be identified in an efficient manner. However, in order to accomplish this, an approach to continuously update and validate the index of predictions would have to be developed.

Hybrid context prediction models. In the previous chapters, we have concentrated on the prediction of discrete context data. With a discrete context model, meaningful changes in the user's context can be predicted, e.g., transitions between places such as home or work. However, in some cases the discrete context model cannot be effectively applied to capture the user's current situation. For instance, while travelling on the road network when driving a car, discrete context information cannot reflect the user's dynamic position. Instead, geographic positions such as GPS coordinates are often employed in this case, so that various prediction techniques are required, either for discrete or continuous context data, to make useful forecasts.

Existing research has proposed methods for both data categories: While the approach presented in this thesis can be used for the prediction of discrete context, techniques such as dead-reckoning have been applied to predict the positions of moving object. However, in current research, those prediction models are always considered independent from each other. For a universally applicable prediction system, both approaches have to be integrated into a powerful hybrid prediction model. Also, the hybrid prediction model could exploit inherent interdependencies among both approaches to improve the prediction accuracy. For instance, assuming that a user is driving to his office, the dead-reckoning techniques could be enhanced by including the knowledge about the predicted trip destination. This can help to further increase the prediction accuracy, as the future movement trajectory computed by the dead-reckoning algorithm is able to incorporate the most probable destinations targeted by the user.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AR	Autoregressive Model
ARMA	Autoregressive Moving Average Model
BN	Bayesian Network
DBN	Dynamic Bayesian Network
CMDP	Constrained Markov Decision Process
CSL	Continuous Stochastic Logic
CTL	Computational Tree Logic
DP	Dynamic Programming
GPS	Global Positioning System
GPRS	General Packet Radio Service
HMM	Hidden Markov Model
LP	Linear Programming
MA	Moving Average Model
MDP	Markov Decision Process
NLP	Non-Linear Programming
SMM	Semi-Markov Model
UMTS	Universal Mobile Telecommunications System
WGS84	World Geodetic System 1984

Bibliography

[AAH+09] Theodoros Anagnostopoulos, Christos Anagnostopoulos, Stathes Hadjiefthymiades, Miltos Kyriakakos, and Alexandros Kalousis. Predicting the Location of Mobile Users: A Machine Learning Approach. In *Proceedings of the 6th International Conference on Pervasive Services (ICPS)*, 2009.

[AG02] U-Blox AG. GPS Navigation Performance of TIM GPS Receivers. http://www.u-blox.com, April 2002.

[Alt99] Eitan Altman. *Constrained Markov Decision Process*. Chapman & Hall/CRC, 1999.

[Ama11] Amatriain, Xavier and Jaimes, Alejandro and Oliver, Nuria and Pujol, Josep M. Data Mining Methods for Recommender Systems. In *Recommender Systems Handbook*. Springer, 2011.

[AR11] J.K. Aggarwal and M.S. Ryoo. Human activity analysis: A Review. *ACM Computing Surveys (CSUR)*, 43:1–43, 2011.

[AS02] D. Ashbrook and T. Starner. Learning Significant Locations and Predicting User Movement with GPS. In *Proceedings of the Sixth International Symposium on Wearable Computers (ISWC)*, 2002.

[Bas00] Basilio Sierra and Iñaki Inza and Pedro Larrañaga. Medical bayes networks. In *First International Symposium on Medical Data Analysis (ISMDA)*, 2000.

[BBHS03] Martin Bauer, Christian Becker, Jörg Hähner, and Gregor Schiele. ContextCube - Providing Context Information Ubiquitously. In *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCS 2003 Workshops)*, 2003.

[BBV09] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference (IMC)*, 2009.

[BD99] Amiya Bhattacharya and Sajal K. Das. LeZi-Update: An Information-theoretic Approach to Track Mobile Users in PCS networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[BDR12a] Patrick Baier, Frank Duerr, and Kurt Rothermel. TOMP: Opportunistic Traffic Offloading Using Movement Predictions. In *Proceedings of the 37th IEEE Conference on Local Computer Networks (LCN)*, 2012.

[BDR12b] Patrick Baier, Frank Dürr, and Kurt Rothermel. PSense: Reducing Energy Consumption in Public Sensing Systems. In *Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 2012.

[Bel57] Richard Bellmann. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.

[BHHK03] Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-Checking Algorithms for Continuous-Time Markov Chains. *IEEE Transactions on Software Engineering*, 29:524–541, 2003.

[BI98] A. F. Bobick and Y. A. Ivanov. Action Recognition Using Probabilistic Parsing. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1998.

[BI04] Ling Bao and Stephen S. Intille. Activity Recognition from User-Annotated Acceleration Data. *Pervasive Computing*, 3001:1–17, 2004.

[BJR08] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2008.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BKH99] Christel Baier, Joost-Pieter Katoen, and Holger Hermanns. Approximate Symbolic Model Checking of Continuous-Time Markov Chains. In *Proceedings of the 10th International Conference on Concurrency Theory (CONCUR)*, 1999.

[BKVR10] Andreas Benzing, Boris Koldehofe, Marco Völz, and Kurt Rothermel. Multi-level Predictions for the Aggregation of Data in Global Sensor Networks. In *Proceedings of the 14th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, 2010.

[BMR04] Suanne Bürklen, Pedro José Marrón, and Kurt Rothermel. An Enhanced Hoarding Approach Based on Graph Analysis. In *Proceedings of the IEEE International Conference on Mobile Data Management (MDM)*, 2004.

[BNKS94] Amotz Bar-Noy, Ilan Kessler, and Moshe Sidi. Mobile Users: to Update or not to Update? In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 1994.

[BNSW94] Norman Adams Bill N. Schilit and Roy Want. Context-aware Computing Applications. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1994.

[BPE07] Web Services Business Process Execution Language Version 2.0, March 2007.

[BZ10] Andrey Boytsov and Arkady B. Zaslavsky. Context Prediction in Pervasive Computing Systems: Achievements and Challenges. In *Supporting Real Time Decision-Making*, Annals of Information Systems. Springer, 2010.

[CEL+08] Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, Ronald A. Peterson, Hong Lu, Xiao Zheng, Mirco Musolesi, Kristóf Fodor, and Gahng-Seop Ahn. The Rise of People-Centric Sensing. *IEEE Internet Computing*, 12:12–21, 2008.

[CGS+09] Ionut Constandache, Shravan Gaonkar, Matt Sayler, Romit Roy Choudhury, and Landon Cox. Energy-efficient Localization Via Personal Mobility Profiling. In *In Proceedings of the the First Annual International Conference on Mobile Computing, Applications, and Services (MobiCase)*, 2009.

[Dey00] Anind K. Dey. *Providing Architetcural Support for Building Context-aware Applications*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, November 2000.

[DH98] Brian D. Davison and Haym Hirsh. Predicting Sequences of User Actions. In *Workshop on Predicting the Future: AI Approaches to Time Series Analysis*, 1998.

[DH07] Elizabeth M. Daly and Mads Haahr. Social Network Analysis for Routing in Disconnected Delay-tolerant MANETs. In *Proceedings of the 8th ACM international Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2007.

[DR03] Frank Dürr and Kurt Rothermel. On a Location Model for Fine-Grained Geocast. In *Proceedings of the Fifth International. Conference on Ubiquitous Computing (UbiComp)*, 2003.

[DS98] Norman R. Draper and Harry Smith. *Applied Regression Analysis*. Wiley-Blackwell, 1998.

[EFH+09] Hanna Eberle, Stefan Föll, Klaus Herrmann, Frank Leymann, Annapaola Marconi, Tobias Unger, and Hannes Wolf. Enforcement from the Inside: Improving Quality of Business in Process Management. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2009.

[EPT+06] Antti J. Eronen, Vesa T. Peltonen, Juha T. Tuomi, Anssi P. Klapuri, Seppo Fagerlund, Timo Sorsa, Gaëtan Lorho, and Jyri Huopaniemi. Audio-based Context Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 14:321–329, 2006.

[FBHR12] Stefan Föll, Florian Berg, Klaus Herrmann, and Kurt Rothermel. A Predictive Protocol for Mobile Context Updates with Hard Energy Constraints. In *Proceedings of the 13th International Conference on Mobile Data Management (MDM)*, 2012.

[FHH10] Stefan Föll, Klaus Herrmann, and Chistian Hiesinger. Flow-Based Context Prediction. In *Proceedings of the 7th International Conference on Pervasive Services (ICPS)*, 2010.

[FHR11] Stefan Föll, Klaus Herrmann, and Kurt Rothermel. PreCon - Expressive Context Prediction using Stochastic Model Checking. In *Proceedings of the 8th International Conference on Ubiquitous Intelligence and Computing (UIC)*, 2011.

[FHR12] Stefan Föll, Klaus Herrmann, and Kurt Rothermel. Energy-efficient Update Protocols for Mobile User Context. In *Proceedings of the 26th IEEE Intl. Conf. on Advanced Information Networking and Applications (AINA)*, 2012.

[FMK+10] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in Smartphone Usage. In *Proceedings of the 8th International Conference on Mobile systems, Applications, and Services (MobiSys)*, 2010.

[GC07] Karthik Gopalratnam and Diane J. Cook. Online Sequential Prediction via Incremental Parsing: The Active LeZi Algorithm. *IEEE Intelligent Systems*, 22:52–58, 2007.

[HCY08] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble Rap: Social-based Forwarding in Delay Tolerant Networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2008.

[HHK+10] Bo Han, Pan Hui, V.S. Anil Kumar, Madhav V. Marathe, Guanhong Pei, and Aravind Srinivasan. Cellular Traffic Offloading Through Opportunistic Communications: a Case Study. In *Proceedings of the 5th ACM workshop on challenged networks (CHANTS)*, 2010.

[HLA+04] Lonnie Harvel, Ling Liu, Gregory D. Abowd, Yu xi Lim, Chris Scheibe, and Chris Chatham. Context Cube: Flexible and Effective Manipulation of Sensed Context Data. In *Proceedings of the Second International Conference on Pervasive Computing (Pervasive)*, 2004.

[How71a] Ronald A. Howard. *Dynamic Probabilistic Systems: Markov Models*. John Wiley & Sons, 1971.

[How71b] Ronald A. Howard. *Dynamic Probabilistic Systems: Semi-Markov and Decision Processes*. John Wiley & Sons, 1971.

[HRKD08] Klaus Herrmann, Kurt Rothermel, Gerd Kortuem, and Naranker Dulay. Adaptable Pervasive Flows - An Emerging Technology for Pervasive Adaptation. In *Proceedings of the Workshop on Pervasive Adaptation at the 2nd International Conference on Self-Adaptive and Self-Organizing Systems*, 2008.

[HS07] M. Hartmann and D. Schreiber. Prediction Algorithms for User Actions. In *In Proceedings of International Conference on Adaptive Business Information Systems (BIS)*, 2007.

[HWR11] Klaus Herrmann Hannes Wolf and Kurt Rothermel. FlexCon: Robust Context Handling in Human-Oriented Pervasive Flows. In *Poceedings of the 19th International Conference on Confederated International Conferences (CoopIS)*, 2011.

[KH05] David Kotz and Tristan Henderson. CRAWDAD: A Community Resource for Archiving Wireless Data at Dartmouth. *IEEE Pervasive Computing*, 4(4):12–14, oct 2005.

[KLGT09] Mikkel Baun Kjaergaard, Jakob Langdal, Torben Godsk, and Thomas Toftkjaer. EnTracked: Energy-efficient Robust Position Tracking for Mobile Devices. In *Proceedings of the 7th International Conference on Mobile systems, Applications, and Services (MobiSys)*, 2009.

[KM09] Dimitrios Katsaros and Yannis Manolopoulos. Prediction in Wireless Networks by Markov Chains. *IEEE Wireless Communications*, 16:56–63, 2009.

[KNP07] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic Model Checking. *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation*, 4486:220–270, 2007.

[Kop08] Daniel; Wutke Daniel; Leymann Frank Kopp, Oliver; Martin. On the Choice Between Graph-Based and Block-Structured Business Process Modeling Languages. In *Modellierung betrieblicher Informationssysteme (MobIS 2008)*, 2008.

[KR01a] Uwe Kubach and Kurt Rothermel. A Map-Based Hoarding Mechanism for Location-Dependent Information. In *Proceedings of the Second International Conference on Mobile Data Management (MDM)*, 2001.

[KR01b] Uwe Kubach and Kurt Rothermel. Exploiting Location Information for Infostation-based Hoarding. In *Proceedings of the 7th Annual International Conference on Mobile computing and Networking (MobiCom)*, 2001.

[KR02] Uwe Kubach and Kurt Rothermel. Estimating the Benefit of Location-Awareness for Mobile Data Management Mechanisms. In *Proceedings of the International Conference on Pervasive Computing (Pervasive)*, 2002.

[Kru11] John Krumm. Ubiquitous Advertising: The Killer Application for the 21st Century. *Pervasive Computing*, 10(1):66–73, 2011.

[KWK+09] Kai Kunze, Florian Wagner, Ersun Kartal, Ernesto Morales Kluge, and Paul Lukowicz. Does Context Matter? A Quantitative Evaluation in a Real World Maintenance Scenario. In *Proceedings of the 7th International Conference on Pervasive Computing (Pervasive)*, 2009.

[KWM10] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity Recognition using Cell Phone Accelerometers. *ACM SIGKDD Explorations Newsletter*, 12:74–82, 2010.

[LDR08] Ralph Lange, Frank Dürr, and Kurt Rothermel. Scalable Processing of Trajectory-Based Queries in Space-Partitioned Moving Objects Databases. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2008.

[LFC06] Jérémie Leguay, Timur Friedman, and Vania Conan. Evaluating Mobility Pattern Space Routing for DTNs. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[LH06] Jong-Known Lee and Jennifer C. Hou. Modeling Steady-state and Transient Behaviours of User Mobility: Formulation, Analysis, and Application. In *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2006.

[LHK01] Gabriel G. Infante Lopez, Holger Hermanns, and Joost-Pieter Katoen. Beyond Memoryless Distributions: Model Checking Semi-Markov Chains. In *Process Algebra and Probabilistic Methods. Performance Modeling and Verification*, 2001.

[LML+10] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. A Survey of Mobile Phone Sensing. *IEEE Communications Magazine*, 48:140–150, 2010.

[LPL+09] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. SoundSense: Scalable Sound Sensing for People-centric Applications on Mobile Phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys)*, 2009.

[LR01] Alexander Leonhardi and Kurt Rothermel. A Comparison of Protocols for Updating Location Information. *Cluster Computing*, 4(4):355–367, October 2001.

[LRC+12] Hong Lu, Mashfiqui Rabbi, Gokul T. Chittaranjan, Denise Frauendorfer, Marianne Schmid Mast, Andrew T. Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. StressSense: Detecting Stress in Unconstrained Acoustic Environments using Smartphones. In *Proceedings of the 14th International Conference on Ubiquitous Computing (Ubicomp 2012)*, 2012.

[Mac67] James B. MacQueen. Some Methods for Classiffcation and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[May04] Rene Michael Mayrhofer. *An Architecture for Context Prediction*. PhD thesis, Johannes Kepler University of Linz, 2004.

[May05] Rene Mayrhofer. Context Prediction based on Context Histories: Expected Benefits, Issues and Current State-of-the-Art. In *Proceedings of the 1st International Workshop on Exploiting Context Histories in Smart Environments*, 2005.

[Men02] Scott W. Menard. *Applied Logistic Regression*. Sage Publications, 2002.

[MLF+08] Emiliano Miluzzo, Nicholas D. Lane, Kristof Fodor, Ronald A. Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application. In *Proceedings of 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2008.

[MM07] Justin Muncaster and Yunqian Ma. Activity Recognition using Dynamic Bayesian Networks with Automatic State Selection . In *Proceedings of the IEEE Workshop on Motion and Video Computing (WMVC)*, 2007.

[MPF+10] Mirco Musolesi, Mattia Piraccini, Kristof Fodor, Antonio Corradi, and Andrew T. Campbell. Supporting Energy-Efficient Uploading Strategies for Continuous Sensing Applications on Mobile Phones. In *Proceedings of the 8th International Conference on Pervasive Computing (Pervasive)*, 2010.

[MPS+09] Annapaola Marconi, Marco Pistore, Adina Sirbu, Hanna Eberle, Frank Leymann, and Tobias Unger. Enabling Adaptation of Pervasive Flows: Built-in Contextual Adaptation. In *Proceedings of the 7th International Joint Conference ICSOC-Service Wave*, 2009.

[MPV01] Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to Linear Regression Analysis*. Wiley, 2001.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[MSA03] Arunesh Mishra, Minho Shin, and William Arbaugh. An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process. *ACM SIGCOMM Computer Communication Review*, 33:93 – 102, 2003.

[MSA04] Arunesh Mishra, Minho Shin, and William Arbaugh. Context Caching using Neighbor Graphs for Fast Handoffs in a Wireless Network. In *23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[Mur02] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[Nee12] Neelamadhab Padhy and Pragnyaban Mishra and Rasmita Panigrahi. The Survey of Data Mining Applications And Feature Scope. *Asian Journal of Computer Science & Information Technology*, 2:68–77, 2012.

[NSMP11] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. An Empirical Study of Geographic User Activity Patterns in Foursquare. In *Proceedings of Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 2011)*, 2011.

[Pea85] Judea Pearl. Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society*, 1985.

[PNF05] M. Martin P. Nurmi and J. A. Flanagan. Enabling Proactiveness through Context Prediction. In *In Proceedings of the 2nd Workshop on Context Awareness for Proactive Systems*, 2005.

[PP09] Kurt Partridge and Bob Price. Enhancing Mobile Recommender Systems with Activity Inference. In *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP)*, 2009.

[Qui86] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[Rab89] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[RBB+03] Abhishek Roy, Soumya K. Das Bhaumik, Amiya Bhattacharya, Kalyan Basu, Diane J. Cook, and Sajal K. Das. Location Aware Resource Management in Smart Homes. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, 2003.

[RDB07] Abhishek Roy, Sajal K. Das, and Kalyan Basu. A Predictive Framework for Location-Aware Resource Management in Smart Homes. *IEEE Transactions on Mobile Computing*, 6:1270–1283, 2007.

[RH10] Parisa Rashidi and Lawrence B. Holder. Discovering Activities to Recognize and Track in a Smart Environment. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):527–539, 2010.

[RLFC11] Daniel Roggen, Paul Lukowicz, Alois Ferscha, and Ricardo Chavarriaga. The OPPORTUNITY Framework and Data Processing Ecosystem for Opportunistic Activity and Context Recognition. *International Journal of Sensors, Wireless Communications and Control, Special Issue on Autonomic and Opportunistic Communications*, 2011.

[RMD07] A. Roy, A. Misra, and S.K. Das. Location update versus paging trade-off in cellular networks: An approach based on vector quantization. *IEEE Transactions on Mobile Computing*, 6:1426–1440, 2007.

[RMM+10] Kiran K. Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter J. Rentfrow, Chris Longworth, and Andrius Aucinas. EmotionSense: a Mobile Phones based Adaptive Platform for Experimental Social Psychology Research. In *Proceedings of the 12th ACM International Conference on Ubiquitous computing (Ubicomp)*, 2010.

[RMM+11] Kiran K. Rachuri, Cecilia Mascolo, Mirco Musolesi, , and Peter J. Rentfrow. SociableSense: Exploring the Trade-offs of Adaptive sampling and Computation Offloading for Social Sensing. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2011.

[RN09] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2009.

[RRD06] Nirmalya Roy, Abhishek Roy, and Soumya K. Das. Context-Aware Resource Management in Multi-Inhabitant Smart Homes: A Nash H-Learning based Approach. In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, 2006.

[RZ07] Ahmad Rahmati and Lin Zhong. Context-for-wireless: Context-sensitive energy-efficient wireless data transfer. In *Proceedings of the 5th international Conference on Mobile Systems, Applications and Services (MobiSys)*, 2007.

[SARK09] Marjorie Skubic, Gregory Alexander, Mihail Popescu Marilyn Rantz, and James Keller. A Smart Home Application to Eldercare: Current status and Lessons Learned. *Technology and Health Care - Smart Environments: Technology to Support Healthcare*, 17:183–201, 2009.

[SBG99] Albrecht Schmidt, Michael Beigl, and Hans-Werner Gellersen. There is More to Context Than Location. *Computers & Graphics Journal*, 23:893–902, 1999.

[SG05] Pete Steggles and Stephan Gschwind. The Ubisense Smart Space Platform. In *Proceedings of the Third International Conference on Pervasive Computing (Pervasive)*, 2005.

[SGP+05] W. Schwinger, Ch. Grün, B. Pröll, W. Retschitzegger, and A. Schauerhuber. Context-Awareness in Mobile Tourism Guides - A Comprehensive Survey. 2005.

[Sha05] Shaw, James W. and Horrace, William C. and Vogel, Ronald J. The Determinants of Life Expectancy: An Analysis of the OECD Health Data. *Southern Economic Journal*, 71:768–783, 2005.

[Sig08] Stephan Sigg. *Development of a Novel Context Prediction Algorithm and Analysis of Context Prediction Schemes*. PhD thesis, University of Kassel, 2008.

[SKJH04] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating Next-Cell Predictors with Extensive Wi-Fi Mobility Data. *IEEE Transactions on Mobile Computing*, 5:1633–1649, 2004.

[ST94] Bill N. Schilit and Marvin T. Theimer. Disseminating Active Map Information to Mobile Hosts. *IEEE Network*, 8:22–32, 1994.

[TF08] M. Tentori and J. Favela. Activity-Aware Computing for Healthcare. *IEEE Pervasive Computing Magazine*, 7:51–57, 2008.

[TIL04] E. Munguia Tapia, S. S. Intille, and K. Larson. Activity Recognition in the Home Setting Using Simple and Ubiquitous Sensors. In *Proceedings of the 2nd International Conference on Pervasive Computing*, 2004.

[Tsa02] Ruey S. Tsay. *Analysis of Financial Time Series*. Wiley, 2002.

[WDR10] Harald Weinschrott, Frank Dürr, and Kurt Rothermel. StreamShaper: Coordination Algorithms for Participatory Mobile Urban Sensing. In *Proceedings of the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2010.

[Wei05] William W. S. Wei. *Time Series Analysis: Univariate and Multivariate Methods*. Addison Wesley, 2005.

[WFH11] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2011.

[WGS94] WGS84 Military Standard, 1994.

[WHFG92] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems (TOIS)*, 10:91–102, 1992.

[WKZA10] Yi Wang, Bhaskar Krishnamachari, Qing Zhao, and Murali Annavaram. Markov-optimal Sensing Policy for User State Estimation in Mobile Devices. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2010.

[WLTS06] Jamie A. Ward, Paul Lukowicz, Gerhard Tröster, and Thad Starner. Activity Recognition of Assembly Tasks using Body-worn Microphones and Accelerometers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1553–1567, 2006.

[WMH60] B. Widrow and Jr M.E. Hoff. Adaptive Switching Circuits. In *IRE WESCON Convention Record*, 1960.

[WSCY99] Ouri Wolfson, A. Prasad Sistla, Sam Chamberlain, and Yelena Yesha. Updating and Querying Databases that Track Mobile Units. *Distributed and Parallel Databases*, 7(3):257–287, July 1999.

[ZL78] Jacob Ziv and Abraham Lempel. Compression of Individual Sequences via Variable-rate Coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978.