Institute of Parallel and Distributed Systems
University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3024

Expressive context prediction using Model Checking

Klaus Peter Gruber

Course of Study:	Computer Science
Examiner:	Prof. Dr. Kurt Rothermel
Supervisor:	Dipl.-Inf. Stefan Föll
Commenced:	May 1st, 2010
Completed:	November 2nd, 2010
CR-Classification:	C.2.4, D.2.4, G.3, I.2.6

Contents

1 Abstract **7**

2 Introduction **9**

2.1 Motivation 10

2.2 Use cases 12

2.3 Goals of the thesis 14

2.4 Outline 15

3 Related work **17**

3.1 Classification of context prediction methods 18

3.2 Artificial Neural Networks 18

3.3 Support Vector Machines 20

3.4 Autoregressive Moving Average methods 21

3.5 Markov Models 21

3.6 Hidden Markov Models 22

3.7 Bayesan Networks 23

4 Requirement analysis **25**

4.1 Prediction quality 26

4.2 Uncertainty 27

4.2.1 Uncertainty in behavior 27

4.2.2 Uncertainty in recognition 27

4.3 Proactivity 28

4.4 Adaptive unsupervised learning at run-time 28

4.5 Incremental model growth 29

4.6 Querying interface 30

4.6.1 Active queries 30

4.6.2 Continuous queries 30

4.7 Temporal prediction 31

4.8 Expressiveness and dynamics 32

4.9 Limited use of resources 32

4.10 Summary and discusssion 33

5 Theoretical background **35**

5.1 Model Checking 36

5.1.1 Classical Model Checking 38

5.1.2 Probabilistic Model Checking 38

5.1.3 Discussion 39

5.2 Markov chains 40

5.2.1 Discrete time Markov chains 40

5.2.2 Continuous time Markov chains 41

5.2.3 Semi-Markov chains 42

5.2.4 Discussion 44

6 Context prediction using Model Checking **47**

6.1 Definition of Context Model 48

6.1.1 Discrete context 49

6.2 Definition of State Transition System 50

6.2.1 Formal model 50

6.2.2 Example scenario 51

6.3 Learning of probabilistic state transition system 53

6.3.1 Learning of transition system 54

6.3.2 Learning of transition probabilities 54

6.3.3 Learning of transition times 55

6.3.4 Ageing 56

6.3.5 Incorporation of uncertainty in states 60

6.4 Definition of query interface 61

6.4.1 Active queries 61

6.4.2 Continuous queries 61

6.5 Query specifications and semantics 62

6.5.1 $X_{P \cong_p}(\varphi)$ 62

6.5.2 $\hat{X}$ . . . 62

6.5.3 $X_{P\cong p}^{T\geq t}(\varphi)$ . . . 63

6.5.4 $\hat{X}^{T\geq t}$ . . . 63

6.5.5 $X_{P\cong p}^{t\leq T\leq t+\Delta t}(\varphi)$ . . . 64

6.5.6 $\hat{X}^{t\leq T\leq t+\Delta t}$ . . . 64

6.5.7 $\Phi U_{P\cong p}^{t\leq T\leq t+\Delta t}\Psi$ . . . 65

6.5.8 $X_{P\cong p}^{-1}(\varphi)$ . . . 66

6.6 Computation of context predictions . . . 67

6.6.1 Model Checking algorithms for Semi-Markov Processes . . . 67

6.6.2 $X_{P\cong p}(\varphi)$ . . . 67

6.6.3 $X_{P\cong p}^{T\geq t}(\varphi)$ . . . 68

6.6.4 $X_{P\cong p}^{t\leq T\leq t+\Delta t}(\varphi)$ . . . 70

6.6.5 $\Phi U_{P\cong p}^{t\leq T\leq t}\Psi$ . . . 72

6.6.6 $X^{-1}(p,\varphi)$ . . . 76

6.6.7 Prediction of formula satisfaction time . . . 79

6.6.8 Incorporation of uncertainty in states . . . 79

6.6.9 Optimizations to existing Model Checking algorithms . . . 80

7 Evaluation **83**

7.1 General approach . . . 84

7.1.1 Generation of random user traces . . . 84

7.1.2 Roulette-wheel selection . . . 85

7.2 Performance metrics . . . 87

7.3 Real world data sets . . . 90

7.3.1 The Mainkofen hospital data set . . . 90

7.3.2 The Dartmouth wireless data set . . . 92

7.4 Evaluation of transition systems . . . 93

7.4.1 State space . . . 93

7.4.2 State properties . . . 95

7.4.3 Transition properties . . . 99

7.4.4 Conclusion . . . 104

7.5 Evaluation of operators . . . 106

7.5.1 $X(\varphi)$ . . . 106

7.5.2 $\hat{X}$ . . . 111

7.5.3 $X^{t \geq T}(\varphi)$ . . . 115

7.5.4 $\hat{X}^{t \geq T}$ . . . 118

7.5.5 $X^{T \leq t \leq T+\Delta T}(\varphi)$ . . . 122

7.5.6 $\hat{X}^{T \leq t \leq T+\Delta T}$ . . . 125

7.5.7 $X^{-1}(\varphi)$ . . . 129

7.5.8 $\Phi U^{T \leq t \leq T+\Delta T} \Psi$ . . . 131

7.6 Results . . . 136

7.6.1 Effects of probability thresholds . . . 136

7.6.2 Effects of time window sizes . . . 137

7.6.3 Necessary granularity for time intervals 138

7.6.4 Semi-Markov chains vs continuous time Markov chains 138

7.6.5 Semi-Markov chains vs discrete time Markov chains 139

8 Conclusions **149**

9 Future work and open issues **151**

9.1 Investigate the effects of using k-order Markov Chains 151

9.2 Prediction of hierarchical context 151

9.3 Symbolic Model Checking using PLTL 152

9.4 Ageing and uncertainty evaluation 152

10 Appendix **153**

Bibliography 155

List of figures 159

List of algorithms 163

Chapter 1

Abstract

In today's world the sensors of mobile devices increasingly allow applications for both sensing and reacting based on the surrounding environment. One very promising application of sensor data is the prediction of future contexts. But today's existing technologies for context prediction often are rather computationally expensive and not very well suited to the requirements of this new field.

Research in the field of context prediction is currently limited to predicting the next possible contexts in time that a user may enter. So more expressive predictions that deal with complex behaviour in future context are currently not yet being dealt with. This thesis aims to extend these limits and to enable more complex predictions by using Model Checking methods. Model checking is an efficient tool to validate systems against formal specifications.

In this thesis context prediction is being treated as a probabilistic Model Checking problem. In order to accomplish this we formulate queries in temporal logics that allow predictions about future contexts. These queries formulated in temporal logics will then be computed using a learned behaviour model of a user and evaluated using probabilistic methods.

We evaluate the results of these predictions using two real world data sets. The results show that it is worthwhile to use Model Checking methods for context prediction.

Chapter 2

Introduction

In this chapter we give an introduction into the whole subject of context prediction. In section 2.1 we motivate why context prediction is an important field, especially for pervasive environments. We also present several use cases for pervasive applications that facilitate context prediction to the reader in section 2.2. Section 2.3 contains the goals of this thesis and section 2.4 introduces our approach of accomplishing these goals and the further structure of this thesis.

2.1 Motivation

According to Wikipedia [1] ubiquitous or pervasive computing is defined as follows:

> "Ubiquitous computing is a model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities. In the course of ordinary activities, someone using ubiquitous computing engages many computational devices and systems simultaneously, and may not necessarily even be aware that they are doing so. [...] This paradigm is also described as pervasive computing or ambient intelligence."

In general we can say ubiquitous or pervasive computing is about small, inexpensive networked processing devices, distributed at all scales throughout everyday life. It uses context information to adapt the behavior of software applications to human needs. Even more convenience is reached if the appliance foresees the users desires and acts proactively. Context histories, especially when recorded over a long period of time, offer a wide range of possibilities to enhance the services provided by some computer application. These possibilities include inferring of current and past user actions, selection of devices, etc.

Correctly predicting future contexts based on past context histories is a major challenge in the field of pervasive computing. Being able to correctly predict future contexts allows pervasive applications to put this knowledge to use in order to assist users. Having information about possible future contexts offers advantages over only using past and current contexts for applications. Pervasive applications can use the obtained knowledge in order to perform actions for a user, but it is also possible for pervasive applications to use predicted context without triggering any actions at all. Proactivity allows applications to provide user interaction that conforms better to the user's expectations. A very simple example for proactivity is the prefetching of information based on possible future contexts, making information available to the user in time, before a context actually is being entered. This is just one example for a prerequisite for a significant improvement in user interaction.

People today are increasingly embedded in an environment consisting of uncountable computing devices and artifacts that provide various degrees of computing power and awareness. This even happens without people consciously noticing it. Current estimations

state that ten years from now, there will be 7 billion people living on this planet surrounded by 7 trillion wireless devices and sensors [2].

Pervasive computing is about the interaction between computers and the real world with computers being embedded in almost all real world object one can imaginane. Applications for pervasive environments usually have interfaces called sensors and actuators. The sensors provide access for the pervasive application to gain knowledge about the real world and to apply this knowledge to the virtual world. The actuators are optional and used in order to apply actual virtual knowledge to the real world, e.g. to control physical properties of objects. Usually pervasive applications run in a massively distributed fashion on computing devices being networked together with wireless network technology in order to allow for mobile use.

The idea behind pervasive applications is the support of mobile human users in their daily activities. There are several challenges to take in order to make this idea come true. They include the definition of adequate hardware architectures and communication protocols, but also includes aspects of software distribution. The most challenging problem in pervasive computing is how a pervasive application can adapt to the user in order to support him in an unobtrusive way.

Ideally, the application should run in the background unnoticed by the user and adapt to his actions. Today, the interaction with computer systems of any kind is mostly still done explicitly and manually in most cases. However, this is not an option for pervasive systems that shall offer support to a wide spectrum of users that may not be capable of dealing with the specificities of computing devices, even more so as these devices may not even be visible. This is why further research into the field of context prediction with a focus on pervasive environments is required.

2.2 Use cases

In order to demonstrate the use of context prediction we present two example use cases to the reader. These examples illustrate the benefits for users when using pervasive computing applications that perform context prediction.

Smart home In [3] the authors describe a house reacting to its inhabitants' needs like a rational agent. In order for this to work the house needs to be able to predict the inhabitants' mobility patterns and their device usage. The idea behind this is to maximize comfort and productivity for its inhabitants and to minimize the operational costs of the household, e.g. by saving energy. This can be accomplished by heating up the bathroom right in time in the morning some time before the first inhabitant usually takes a shower. This way less energy is wasted and the inhabitants don't have to worry about setting their alarms to 30 minutes before they actually need to get up in order to turn up the heating. After the bathroom has been used by all inhabitants the heating can be turned off again. In a similar way comfort can be increased by the coffee brewing machine starting to brew coffee at certain times during the day, saving the inhabitants' time. A further idea would be a regridgerator that is aware of the groceries stored inside and that automatically orders new groceries when the current stock gets low. When used in combination with context prediction the refridgerator could detect patterns and always order the right amount of groceries in advance, e.g. for a big family breakfast on sundays.

Intelligent mobile devices Nowadays smart phones get more and more popular especially with young people. Almost all people also have internet access with their smart phones along in their mobile plans. Usually this is achieved by the mobile providers by using UMTS with a semi-flatrate that allows the full UMTS speed to be used until a certain quota has been reached. After the quota has been reached the speed is usually reduced to GPRS which is significantly slower. In times of youtube thought this quota can easily be reached very fast when not taking care.

One idea to aid the user in not wasting his quota is to delay certain activities until a wireless network becomes available. This way the quota will not be reached as fast. For example given the situation of sending a big email of several megabytes in size the smart

phone could decide to delay the sending via UMTS or GPRS if the previous user behaviour suggests that it is very likely that the user will have wireless access with his phone within the next half an hour. This would conserve quota for the user or speed up the sending process in case the quota has already been reached. Of course the user would also have to assign a priority in order to force an email to be sent right away. In order to do this a location predictor would be needed that can predict the future location of the user with a high accuracy.

2.3 Goals of the thesis

Research in the field of context prediction is limited to predicting the next possible contexts in time that a user may enter. So more expressive predictions that deal with complex behaviour in future contexts are currently not yet being dealt with. This thesis aims to extend these limits and to enable more complex predictions by using Model Checking methods. Model checking is an efficient tool to validate systems against formal specifications.

In this thesis context prediction is being treated as a probabilistic Model Checking problem. In order to accomplish this we formulate queries in temporal logics that allow predictions about future contexts. These queries formulated in temporal logics will then be computed using a learned behaviour model of a user and evaluated using probabilistic methods.

In order to accomplish this we solve a series of challenges that do not come up when using Model Checking in its classic form. In classic Model checking the system model in use is always static. So changes occuring during run time can not be taken into account during run time using the classical approach. But in context prediction a change in the current context of a user always involves a change to the system model, making the system model highly dynamic. In this thesis we develop different means to adapt the behaviour model by using learning algorithms, therefore enabling Model Checking on dynamic system models.

Also in classic Model Checking queries in temporal logics are evaluated only once for checking system properties. Continuous queries that are answered as soon as a prediction comes true are not part of already existing Model checking algorithms. In this thesis we expand existing Model checking algorithms by enabling continuous queries.

Finally we develop a prototype implementation of our work in context prediction using Java which we can use to evaluate our work. Eventually we discuss the results of our work.

2.4 Outline

In chapter 3 we give an overview over the related work in the field of context prediction. We discuss the properties, requirements, advantages and disadvantages of current context prediction approaches.

In chapter 4 we perform a requirements analysis. We work out requirements for a context predictor that extends the functionality of current approaches and enables more expressive queries. This includes factors of uncertainty, learning, active and continuous queries and temporal prediction.

In chapter 5 we give a brief introduction into Model Checking in general and introduce the different basic kinds of Markov Chains that can be used for Model Checking. We will specifically go into detail about Semi-Markov chains.

In chapter 6 we present our approach of applying model checking methods on Markov Chains, based on the beforehand established requirement analysis in chapter 4. Part of this is the learning applied to probabilistic transition systems, the definition of a query interface, the mapping of user traces to Semi-Markov chains and the computation of context predictions for active and continuous queries.

In chapter 7 we introduce the two real world data sets we evaluate using our approach. We explain the metrics we want to use, discuss the properties of the real world data sets and present our evaluation results for the operators we defined in chapter 6.

In chapter 8 we sum up what we achieved using our new approach to context prediction using Model Checking on Semi-Markov chains.

Chapter 9 describes more ideas that came up during research on our subject and which the time frame did not allow to pursue. These ideas could be looked into for future research on this topic.

Chapter 3

Related work

This chapter deals with current related work in the field of context prediction. In section 3.1 we introduce several methods currently being used for context prediction. In the sections that follow we introduce the methods being used by current research in the field.

3.1 Classification of context prediction methods

First of all we we start by classifying context prediction methods into two basic areas:

Continuous time series prediction Continuous time series prediction as defined in [4] tries to find a model of a system by observing its output in terms of time series. The prediction problem can be defined as a mapping from a number of past samples of the time series to one future sample value, typically with discrete time steps. In time series analysis, it is generally assumed that the input data includes a systematical component and random noise. This noise complicates the identification of the patterns and it is thus usually desirable to filter it out before analyzing the data. In the following, only a selection of methods is presented.

Categorical Time Series Prediction The approach of using continuous time series prediction independently on each of the context class degrees of membership has a significant disadvantage. Although some high-level contexts might be statistically independent, most are presumed to be dependent. An example would be the two mutually exclusive contexts, "giving a presentation" and "in a concert". The knowledge or strictly speaking the estimation with a high confidence that one of the contexts is active is a strong indication that the other one is not. We expect people to be engaged in only one high-level context during most of the time, assuming a strong mutual dependency between all context classes. This favors the second option for prediction, namely categorical time series prediction as defined in [4], where the trajectory of the best matching context classes is considered. In the following, we examine single-dimensional categorical time series that only take the best matching context for each time step into account.

3.2 Artificial Neural Networks

One approach using continuous time series prediction are Artificial Neural Networks. Artificial Neural Networks are a family of methods following the principle of combining multiple simple neurons into a network structure to realize a desired behavior. An artificial

neuron is a simple unit which computes a linear, weighted sum with an additional output function.

The most important neural network type is the Multi-Layer Perceptron with a strict feed-forward architecture of three layers: the input layer is defined to assume the values of the input vector and does not perform any further computation, the hidden layer is fully connected to the input layer and usually uses the sigmoid function as output function, and the output layer with a number of neurons corresponding to the dimensionality of the output vectors is again fully connected to the hidden layer and applies a linear output function. Such a MLP can be shown to be a universal function approximator, i.e. to be able to represent arbitrary, multi-dimensional functions.

The aim for a Multi-Layer Perceptron generally is to learn the correct mapping from input to output vectors given a training data set of corresponding inputs and outputs. As the network structure and the involved output functions are typically defined a priori, learning the mapping accounts to finding a combination of weights for the neurons in the hidden and the output layers that realizes the desired function. The well-known backpropagation learning algorithm solves this problem for Multi-Layer Perceptrons. This approach has the advantage that the exact function does not need to be known nor does it need to be expressible in a closed mathematical form, because it can be learned from a given training data set, but it has the disadvantage that the internal behavior of a neural network can not be analyzed analytically. It can only be simulated.

For finding a corresponding output vector to a given input the outputs of the hidden layer neurons have to be calculated and subsequently also of the output neurons, which makes reasoning about the weight representation difficult.

The main issue of neural networks and more specifically Multi-Layer Perceptrons is that they are also not suitable for adaptive, online operation and thus not appropriate for embedded systems. Because the network structure needs to be determined a priori, they fail the requirement of incremental model growth and because the back-propagation learning algorithm is a batch algorithm by design, they violate the requirement of online learning. In typical experiments, back-propagation needs hundreds of learning periods to achieve good results. Thus, online learning with only one period for learning does not seem reasonable for this type of neural network [5].

3.3 Support Vector Machines

Another approach for context prediction using continuous time series prediction is by using Support Vector Machines. Support Vector Machines are a newer technique for machine learning and are suitable for both pattern recognition and regression estimation, which can be applied to time series prediction. The basic concept of SVMs is that data that is non-separable in its original space can be mapped to another space where it is separable by a linear hyperplane. This hyperplane is chosen so that the distance between the classes that should be separated is maximized. In the final solution of the separation problem, the hyperplane is defined only by those data points that lie closest to it, i.e. those points which are nearest to the other class and are consequently called support vectors because changing them would also change the hyperplane and thus the solution of the training. If all other points from the data set were removed or moved within the same class, training would still yield the same hyperplane.

In contrast to ANNs, SVMs always find a global solution and are therefore resistant to complex problems with local minima. They also successfully address overfitting, which is a serious problem for almost all methods from the family of ANNs. Although SVMs have started over two decades ago, they currently receive increasing attention because they have been shown to yield high accuracy and outperform competing methods for many problems, including time series prediction.

As in many machine learning techniques, most of the practical problems can not be solved analytically but need to be dealt with by numerical optimization. SVM training generally depends on constrained quadratic optimization and, with current algorithms, SVMs bring forth extreme demands for processing power and are thus difficult to use for embedded systems. For our purpose of context prediction, SVMs can in principle be used in the same way as ANNs for continuous time series prediction, as there are variants for regression. But there are other variants that could be used to predict the next best matching context class when given the current and possibly past class vectors with degrees of membership, which corresponds to categorical time series prediction [6, 7].

3.4 Autoregressive Moving Average methods

Autoregressive Moving Average methods is probably the best known and most often applied method for stationary, continuous time series prediction. It is a combination of autoregressive and moving average models with discrete, equidistant time steps, consisting of linear terms. The univariate output, describing the predicted next sample, depends on the present and the last inputs of the system and the last outputs of the system. Before analyzing a time series with an Autoregressive Moving Average model, it needs to be stripped from its trend and seasonal components, as Autoregressive Moving Average only deals with stationary data. Although it uses solely linear combinations, it has been applied successfully in a wide number of application areas, e.g. for optimizing distributed time warp simulations by predicting event arrival times. The Durbin-Levison and the Innovations algorithms allow to estimate the Autoregressive and Moving Average parameters of an Autoregressive Moving Average process given a time series. After parameter estimation, it can be used to successively predict the next sample value of the time series. Before the parameters can be found, the orders of the Autoregressive and Moving Average components need to be estimated, which is assisted by the sample and partial ACF and rarely needs to be larger than two for Autoregressive as well as Moving Average, but often depends on expert knowledge and trial-and-error. Thus, ARMA violates our requirements of online learning and incremental model growth, but can still be applied to settings with non-autonomous devices [4].

3.5 Markov Models

For an approach using categorical time series prediction we introduce first order Markov models. For each context, the frequency of each successor, often called transition probability, is recorded and the successor with the highest frequency is predicted for the next step. Such a Markov model could be used to predict the most probable context for the next time step. First order Markov models fulfill all core requirements including model growth; the transition probabilities for a newly created context class can be safely initialized with zero until transitions into this new context occur. The run-time complexity for online training and prediction is still low enough to consume negligible processing power

compared to classification. When the underlying process suggests a dependency between pairs of states that are not directly adjacent in the time series, then the first order Markov models can be extended to not only depend on the current state for predicting the next one, but to take a number of past states into account. This number of states defines the order of the higher order Markov models. Such models are, in contrast to first order Markov models, capable of exploiting relationships between temporally distant states. Subject to their order that defines the maximum distance between states for which a relationship can be recognized, higher order Markov models can in principle account for sequential as well as (relative) periodic patterns in the time series [8, 9, 10].

3.6 Hidden Markov Models

Representing categorical time series prediction too, Hidden Markov Models are to the standard model for time series classification in many areas, e.g. speech recognition, handwritten character recognition, gesture recognition or segmentation of DNA sequences. In context computing, they have been applied at various levels, like recognizing location based on audio and video, task prediction or action recognition. So far we have only considered exact models that record, i.e. count, occurrences of observed transitions and calculate probabilities from those transition frequencies.

Hidden Markov Models are a more powerful model, but do not necessarily produce the same model when trained with identical training data. The reason for this issue is that during the estimation of the model parameters, the typically used method, called forward-backward algorithm, leads to local maxima for this optimization problem. Many practical problems have been found not to be solvable by simple Markov Chain models, where the observations are mapped to states. Hidden Markov Models thus add an additional hidden layer, hence the name, which contains a Markov Chain model that indirectly influences the Hidden Markov Model's output via a separate probability distribution. To detail, a complete Hidden Markov Model consists of a set of states, a state transition probability distribution, an observation symbol probability distribution and an initial state distribution.

When training a Hidden Markov model from a data set, the number of states and the

connections between the states need to be determined a priori and all three probability distributions are adjusted by training on the data. Hidden Markov Models have been applied to many problems in time series analysis because there exist mature methods for parameter estimation and simulation, building on a solid theoretical foundation.

However, the parameter estimation is an optimization problem for which no analytical solution is currently known. The training method is an iterative batch approach and we are currently not aware of any online method that would allow to use HMMs in an online way. Incremental model growth is also not supported, as the number of hidden states and the alphabet of observation symbols need to be defined before training and can not be modified afterwards. The major requirement of unsupervised operation is violated by the need to determine the number of hidden states a priori. These issues restrict the Hidden Markov Model method also to scenarios where an autonomous operation of embedded systems without infrastructure support for recomputation is not strictly necessary [9].

3.7 Bayesan Networks

Bayesian Networks are a superset of Hidden Markov Models, Kalman filtering and other probabilistic models. A Bayesan Network is in general a directed acyclic graph of nodes, which represent the random variables, and edges, which represent the direct influence of one variable on another.

Within this structure, each variable is independent of its non-descendants given its parents, and the network can thus be completely parameterized by specifying local models for each variable. In the case of discrete variables with a finite set of values, a standard representation of these local models are tables with conditional distributions. Due to the graphical representation of probabilistic dependence, Bayesan Networks have been widely successful for problems where prior knowledge exists and can be incorporated into the model.

Additionally, the dependence information in Bayesan Networks has a causal semantics and is thus often intuitive for some applications. There are various approaches for learning the parameters of a Bayesan Network from data, including the possibility to learn the model structure even from incomplete data sets with missing values and to perform an

online update of the structure, which allows Bayesan Networks to fulfill the requirement of online learning. However, the number and the structure of hidden variables used for describing influences on the observed variables also have to be specified a priori, which leads to the same issues in unsupervised operation as present with Hidden Markov Models [4].

Chapter 4

Requirement analysis

In this chapter we provide details about requirements pervasive environments demand from a context predictor. We sum up requirements already discussed in earlier publications [11, 4] and add some new requirements that we think could be useful for context prediction. Finally we discuss which current approaches from related work are fulfilling those requirements in section 4.10.

4.1 Prediction quality

A high quality of predictions is one of the main requirements for any context predictor. A high quality in prediction will also quite naturally lead to the best results for the application using a context predictor.

Of course when talking about prediction in general we need metrics that describe the quality of our predictions. Naturally this also applies to the field of context prediction. Using these metrics we can then reason about the quality of our predictions. Prediction quality should be as high as possible, although there might exist trade offs between the quality of the predictions made and their computational or memory requirements. Also different metrics might contradict each other, leading to a tradeoff in prediction quality.

We can also use these metrics in order to compare our approach to other approaches. This enables us to compare the prediction quality of different approaches, but this of course requires to perform the evaluation on the same data sets.

4.2 Uncertainty

User behaviour and sensor recognition can be uncertain. We propose that a context predictor should be able to model these uncertainties. We go now into detail about the different kinds of uncertainty we expect to come across.

4.2.1 Uncertainty in behavior

User behaviour is not deterministic in the sense that it is not always clear what the user is going to do next. This is obvious, if user behaviour was totally deterministic we would not need the field of context prediction. Context prediction especially applies to problems where user behaviour can be very nondeterministic. Any approach for context prediction has to deal with uncertainty in user behaviour and reason about predictions with uncertainty in mind.

The context predictor should also compute an estimate of the correctness of the predicted context. This estimation can then be used by applications as a confidence measure. This way the applications can determine for themselves if they want to trust the prediction for certain actions. It should also be possible to use confidence estimates for predictions since it is important to supply applications with a measure of how certain a computed prediction is, so that applications can treat uncertain predictions differently than more certain ones.

4.2.2 Uncertainty in recognition

Another factor we have to deal with is the fact that the recognition of the current context is not necessarily correct since sensor data can be erroneous. Uncertainty in recognition can occur when the measured sensor data is blurry or erroneous. Of course when we cannnot fully trust our sensor data we need to model this into our approach for context prediction as well.

An example would be a GPS sensor being used for location estimation. Since GPS sensors in civil applications only have a resolution of about tens of meters a GPS sensor being

used to estimate locations e.g. in a house can often lead to uncertain results. An example would be a huge dining room and other smaller rooms connected to this relatively big room. Standing right in the middle of the dining room will most likely lead to a precise location estimation (i.e. dining room) given the dimensions and location of the room itself. But when moving to one of the other smaller rooms location estimation might not be precise enough with a resolution of tens of meters to be able to give a precise location estimation when about to leave the dining room. So even though you might already be in the smaller kitchen the location estimation could still come up with dining room.

4.3 Proactivity

In content prediction we also need some mechanism to allow for proactivity. It should be possible to automatically get notifications when certain circumstances apply instead of querying in intervals for these circumstances. This reduces the computational load and leads to more accurate notification times, since the notifications actually get sent out when the circumstances apply instead of depending on the query rate. Examples for proactivity are accident prevention, alerting or planning aid [12].

4.4 Adaptive unsupervised learning at run-time

For most applications in pervasive environments it is infeasible to switch between different phases for training and prediction. Batch learning would in most cases not be a benefit for the user. Therefore, the context predictor should continuously perform learning and adapt its parameters during normal operation.

It should be possible for the approach to learn new behaviour adaptively at run-time in an efficient manner. Every action of the user should be fed into the learning component and immediately be reflected for incoming queries to the context predictor. This would enable the system to always respond to queries with the most up to date information. It is vital to do the learning adaptively at run-time. As the users habits will change over time, classes must always adapt to new feature values [6].

The internal model topology and the prediction method parameters need to be estimated automatically without human interaction or explicit definition of input/output behavior by experts. For some methods that depend on an internal structure, this includes not only to find suitable parameters, but also to find the model itself, as far as it is specific to the application area. If it is not possible to find a model structure a priori for arbitrary applications and contexts, then it must be learned while the system is in use.

This requirement is valid both for embedded systems, which typically have only a very limited way of user interaction, and for services implemented within an infrastructure. Even if the recomputation of predictive models was handled by infrastructure components instead of the end user devices themselves, e.g. by a GSM service provider for the mobile phones connected to its network, human supervision of the specific model selection processes would not be possible due to scalability issues. Human resources are generally limited, requiring the use of unsupervised methods.

4.5 Incremental model growth

When new classes are detected in the classification step during run-time, new dimensions will be added to class vectors. The prediction algorithm must be able to incrementally increase its internal model topology without requiring a complete retraining. As mentioned, recreating the internal model from scratch will in many cases be impossible, as the complete context history can not be stored in embedded systems because of memory requirements and for reasons of privacy. It is currently unclear if shrinking of class vectors during run-time is also necessary or if "dead" classes that were not activated for a certain time frame could simply receive a minimum probability estimation for becoming active.

4.6 Querying interface

The approach should also have an easy to use interface for other applications to register queries and callback channels. This can be achieved by employing a query language. In general we see the need for two basic kinds of queries:

4.6.1 Active queries

One possible query type is the so called one shot query. The application queries the context predictor and immediately receives the result of the query as soon as it is computed. The context predictor then forgets the query as well as the result of the query. The application can use the information in its own computations from there on.

Example queries could be:

- What is the probability of the next bus stop being North Prospect street?
- What is the most likely next context in general?
- What is the probability of riding a bike and returning home?

4.6.2 Continuous queries

We also see use cases for a different kind of query: continuous queries. These types of query will be registered with the context predictor via the query interface. They will be reevaluated continuously after a certain time period has passed and on every context switch. The result of the query will be delivered to the querying application via a notification. On registering the query the application has to state the lifetime of the query and when it would like to get a notification, i.e. on every time the query becomes true, only for the first k times, or for a time window of t seconds.

Example queries could be: Send a notification when the probability of going to the bathroom within the next 30 minutes is higher than 0.75 in order to turn on the heating.

4.7 Temporal prediction

Instead of taking time as a discrete parameter we see the need of taking time as a continuous parameter. This means that the model should reflect that spending more time in a context can result in a different follow up context. Also we see the need of temporal queries.

A simple example: During the week days an employee always gets up at 7 AM and goes to work. On Saturday and Sunday though the employee gets up at 10 AM and reads his newspaper and enjoys his spare time.

Now we could want to compute the following query: Given the current context (i.e. the employee is home), what is the probability of going to work for the employee?

It is not possible to model this behaviour if we are not able to take the prediction of time into account for our approach of context prediction.

If we do not model time at all we could only say that according to our data, e.g. 5 out 7 times in a week the employee goes to work and 2 out of 7 times he reads his newspaper. Using this approach for context prediction would therefore result in a high error rate, since for every day we would only guess if it was a week day or a weekend resulting in a static answer for the query.

We could solve this problem by adding the week days into our context model as a seperate discrete context and could therefore reformulate the query: Given the current context (i.e. the employee is home, it's Saturday), what is the probability of going to work?

The probability of the employee working on a Saturday would then be 0, perfectly describing reality. Since we discretize time we soon run into a problem. First of all, granularity of our discretization affects the quality of our results. In the upper example a granularity of 24 hours is enough, but for other areas we might want a granularity of 10 seconds.

Also queries like "At what time will the employee most likely leave home for work" can only be answered if we take time into account.

4.8 Expressiveness and dynamics

We need an expressive model that enables us to reason about different context properties. So for example it should be possible to not only look at location properties but also to look at the activities being performed at the same time, creating multidimensional contexts. When incorporating more than one property into contexts it can again be interesting to reason only about one property, e.g. location instead of all properties being represented in the model.

Expressiveness can also be interpreted as allowing different kinds of queries that go beyond only predicting the next possible state. An example for this would be a query like "What is the probability of taking a taxi in the next 24 hours?" Also it should be possible to nest queries within each other.

4.9 Limited use of resources

In pervasive environments there are usually restrictions in terms of computational power, memory and battery time. We propose that content prediction algorithms should take these aspects into account. Therefore the approach should not require huge amounts of computational power or memory.

4.10 Summary and discusssion

The most important requirements for context prediction in our eyes are temporal prediction, limited use of resources, expressiveness and dynamics, incremental model growth, adaptive unsupervised learning at run-time and of course the prediction quality.

According to our requirements adaptability and online learning are important requirements for a context predictor. However, despite their good general performance for context prediction, Artificial Neural Networs, ARMA methods and Hidden Markov Models are only able to perform batch learning and can not be used to perform online learning. Also incremental model growth is a problem for Artificial Neural Networks and ARMA methods. In addition unsupervised learning can not be performed with SVMs. In our eyes this disqualifies these methods for the use of content prediction in the field of pervasive computing.

Support Vector Machines are another promising approach to context prediction, but the fact that they have very high computational requirements makes SVMs not a good candidate for context prediction in pervasive environments where devices usually have low CPU and battery power. Bayesan Networks are capable of online learning, but they are not capable of unsupervised learning. In a pervasive environment though we can not imagine to have human assistance for the aspect of learning since the sheer amount of incoming data would in most cases ask too much of human interaction. Also none of the approaches above allow for the prediction and querying of multidimenasional context.

This basically leaves us with general Markov Models of 2^{nd} order or higher that do not impose any restrictions on our requirements. We will look into several special classes of Markov Chains. These are deterministic time Markov chains, continuous time Markov chains and Semi-Markov chains. Since our approach will facilitate Model Checking methods in order to perform context prediction we will now give a short introduction into Model Checking.

Chapter 5

Theoretical background

5.1 Model Checking

Model checking as originally defined is a technique for the automatic verification of distributed, concurrent systems [13, 14, 15]. Given a system model and properties of the system as an input, a model checker will compute if these properties hold for this system. In contrast to other methods of testing system properties, in model checking all possible input values are automatically tested. It is a verification technique that is being used to opbtain certain properties of computer software, hardware circuits or communication protocols. When sing Model Checking for the verification process, these properties are specified using a temporal logic like CTL or LTL. Usually systems are represented as transition systems. These properties can be anything from "Is it possible for process A and process B to get into a deadlock situation" to "Is it guaranteed that the user receives a system notification upon a transmission error within 30 seconds?".

The definition of model checking according to [16]:

> "Model checking is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model."

Model checking is a technique for verification of systems that explores all system states as system possibly can enter in a brute-force manner. In this way, it can be shown that a given system model truly satisfies a certain property because the Model checker systematically checked all possible system states. It is a real challenge to examine the largest possible state spaces that can be treated with current processors and memories. State-of-the-art model checkers can handle state spaces of about 10^8 to 10^9 states with explicit state-space enumeration. Using clever algorithms and tailored data structures, even larger state spaces can be handled for specific problems.

Typical properties of qualitative nature are checked using model checking, e.g. Is the generated result correct? Can the system reach a deadlock situation, e.g., when two concurrent programs are waiting for each other and thus halting the entire system? But also timing properties can be checked: "can a deadlock occur within 1 hour after a system reset?" or "is a response always received within 8 minutes?". Model checking requires a precise and unambiguous statement of the properties to be examined. As with making an

accurate system model, this step often leads to the discovery of several ambiguities and inconsistencies in the informal documentation.

The system model is usually automatically generated from a model description. Usually the model description is specified in a programming language. This model description is then converted to the system model using temporal logics like CSL, CTL or LTL. The properties to be ascertained are then also formulated using temporal logics. Note that the property specification prescribes what the system should do, and what it should not do, whereas the model description addresses how the system behaves. The model checker examines all relevant system states to check whether they satisfy the desired property.

Now for a more formal definition of how Model Checking works:

A transition system TS is a tuple $(S, Act, \rightarrow, I, AP, L)$ where

$$S \text{ is a set of states,}$$

$$Act \text{ is a set of actions,}$$

$$\rightarrow \subseteq S \times Act \times S \text{ is a transition relation,}$$

$$I \subseteq S \text{ is a set of initial states,}$$

$$AP \text{ is the set of atomic propositions and}$$

$$L \rightarrow 2^{AP} \text{ is a labeling function.}$$

The labeling function L relates a set $L(s) \in 2^{AP}$ of atomic propositions to any state s. $L(s)$ intuitively stands for exactly those atomic propositions $a \in AP$ which are satisfied by state s. Given that Φ is a propositional logic formula, then s satisfies the formula Φ if the evaluation induced by $L(s)$ makes the formula Φ true; that is:

$$s \models \Phi \text{ iff } L(s) \models \Phi$$

So given a system model and a specification in logic formulas they can be checked for correctness. We refer to [16] for more details about Model Checking in general.

5.1.1 Classical Model Checking

In classical model checking systems are formalized by Kripke structures. The properties of the system are formalized by formulas of temporal logic. Transforming a system to a Kripke structure generally is a two step process. The first step is to transform the system to a first order representation of the system. The second step is to transform this first order representaton of the system to a Kripke structure. We refer to [16] for a definition and additional details. A Kripke structure already very closely resembles a Markov Chain introduced in the next chapter:

$$AP \text{ is a set of atomic propositions,}$$

$$S \text{ is a finite set of states,}$$

$$I \subseteq S \text{ is the subset of initial states,}$$

$$\delta : S \times S \rightarrow \text{[true,false] is the transition relation and}$$

$$L : S \rightarrow 2^{AP} \text{ is the labeling function.}$$

As you can see this represents a transition system with directed transitions. We now introduce stochastic Model Checking which allows for probabilistic reasoning by simply extending the transition relation. Classic Model Checking can be performed using temporal logics like LTL or CTL.

5.1.2 Probabilistic Model Checking

In Probabilistic model checking classical model checking is expanded by the notion of probability. This is achieved by extending the Kripke structure. In classical model checking a transition system is used without probabilities for any of the state transitions, in stochastic model checking Markov Chains are used instead which allow for state transition probabilities:

$$AP \text{ is a set of atomic propositions,}$$

$$S \text{ is a finite set of states,}$$

$$I \subseteq S \text{ is the subset of initial states,}$$

$$\delta : S \times S \to [0,1] \text{ is the transition relation and}$$

$$L : S \to 2^{AP} \text{ is the labeling function.}$$

Notice that the only difference here is the transition relation describing probabilities instead of boolean values. When performing stochastic Model Checking the temporal logics being used usually are PCTL or PLTL. We will not go into further detail at this point and will instead discuss some of the advantages and disadvantages of Model checking [17, 18, 19].

5.1.3 Discussion

Model checking has several strengths. It is a very general approach for system verification. Wen can therefore apply it to a wide range of applications. Examples for these applications include embedded systems, but also software engineering and hardware design. Model checking also supports partial verification. This means that certain properties can be checked individually without the need of specifying all possible properties. This allows the focusing on the most essential properties first. Also no complete requirement specification is needed.It requires neither a high degree of user interaction nor a high degree of expertise. Furthermore, it can be easily integrated in existing development cycles, its learning curve is not very steep and empirical studies indicate that it may lead to shorter development times. It has a sound and mathematical underpinning; it is based on theory of graph algorithms, data structures, and logic.

However, model checking also has several drawbacks. It is mainly appropriate to control-intensive applications and less suited for data intensive applications as data typically ranges over very big domains. Also Model checking is in general not effectively computable for reasoning about abstract data types. It verifies a system model and not the actual system itself. Therefore any result is only as good as the system model constructed. It checks only stated requirements, i.e. if the user fails to state the correct requirements a system should pass there is no guarantee of the system to actually behave in the right way. The validity of properties that are not checked cannot be judged.

Furthermore It suffers from the statespace explosion problem. This means that the number of states needed to model the system accurately may easily exceed the amount of

available computer memory. Despite the development of several very effective methods to combat this problem, models of realistic systems may still be too large to fit in memory. It does not allow checking generalizations: in general, checking systems with an arbitrary number of components, or parameterized systems, cannot be treated. Model checking can, however, suggest results for arbitrary parameters that may be verified using proof assistants.

5.2 Markov chains

Markov chains are a method of describing probabilistic systems. They consist of states and transitions between those states. Markov chains employ the so called Markov property. A stochastic process has the Markov property if the conditional probability of future states of the process depend only upon the present state. That is, given the present, the future does not depend on the past. A process with this property is called a Markov process or Markov chain. Markov chains have already been successfully used for context prediction, e.g. in [9, 10].

5.2.1 Discrete time Markov chains

A discrete time Markov chain describes a system which is in a fully specified state at each step. This is achieved by modeling these states into a transition system with states and transitions between these states. The steps between the states can be interpreted as time steps, but they might as well be interpreted as any other discrete measurement, e.g. like distance.

A discrete time Markov chain M is a tuple (S, P, L) with

$$S \text{ as a finite set of states,}$$

$$P : S \times S \to [0, 1] \text{ as the transition probability matrix}$$

$$L : S \to 2^{AP} \text{ as the labeling function.}$$

Model checking can be performed on discrete time Markov chains using query languages like PCTL [17, 18].

5.2.2 Continuous time Markov chains

Since discrete time Markov chains discretize time there is an extension to this type of Markov Chains called continuous time Markov chains that allows for continuous time queries. Systems that can be described by continuous time Markov chains are characterised by the "memoryless property". This property is caused by the fact that the time already spent in a state is being modeled by negative exponential distributions only. Because of this property the probability of taking a transition from one state to another state is not depending on the amount of time the system has already spent in the current state.

This section recalls the basic concepts of continuous time Markov chains as originally developed by Markov for finite state spaces and Kolmogorov for denumerable and continuous state spaces.

We define a continuous time Markov chain M as a tuple (S, P, R, L) with

$$S \text{ as a finite set of states,}$$

$$P : S \times S \to [0, 1] \text{ as the transition probability matrix}$$

$$R : S \times S \to \mathbb{R}_{\geq 0} \text{ as the rate matrix, and}$$

$$L : S \to 2^{AP} \text{ as the labeling function.}$$

We assume that for any state s, AP contains a combination of atomic propositions ap which is characteristic for s, i.e., $L(s) = ap$ and $L(s') \neq ap$ for any $s' \neq s$.

Intuitively, $R(s, s') > 0$ iff there is a transition from s to s'. Furthermore, $1 - e^{-R(s,s') \cdot t}$ is the probability that the transition s can be triggered within t time units. Thus, the delay of transition s, s' is governed by the exponential distribution with rate R. If $R(s, s') > 0$ for more than one state s', a competition between the transitions originating in s exists, known as a race condition. The probability to move from a nonabsorbing state s to a particular state s' within t time units, i.e., the transition s' wins the race, is given by

$$P(s, s', t) = \frac{R(s, s')}{E(s)} \cdot \left(1 - e^{E(s) \cdot t}\right)$$

with

$$E(s) = \sum_{s' \in S} R(s, s')$$

denoting the total rate at which any transition outgoing from state s is taken.

More precisely, $E(s)$ specifies that the probability of taking a transition outgoing from state s within t time units is t, due to the fact that the minimum of two exponentially distributed random variables is an exponentially distributed random variable with as rate the sum of their rates. Consequently, the probability of moving from a nonabsorbing state s to s' by a single transition, denoted $P(s, s')$ is determined by the probability that the delay of going from s to s' finishes before the delays of other outgoing edges from s. Formally,

$$P(s, s') = \frac{R(s, s')}{E(s)}$$

In order to perform Model Checking on continuous time Markov chains CSL can be used. CSL is a temporal logic like CTL with state and path formulas. CSL is a natural extension to CTL with two operators that allow reasoning about steady-state and transient behavior of a system. Whereas the steady-state operator refers to the probability of residing in a particular set of states in the long run, the transient operator allows us to refer to the probability of the occurrence of particular paths in the continuous time Markov chain. In order to express the time span of a certain path, the path operators $\Phi U \Psi$ and X will be extended with a parameter that specifies a time interval [20, 21].

5.2.3 Semi-Markov chains

Although exponential distributions appropriately model many phenomena occuring in the real world, in many occasions they are inadequate to successfully model the behaviour of a system. For example, human behaviour usually cannot be appropriately modelled [21].

In order to model these phenomena in an adequate manner, general distributions have to be used instead of relying on only exponential distributions. Examples for general distributions include the heavy-tail distribution, the log-normal distribution and normal distributions in general. Therefore as an alternative approach direct model checking of

Semi-Markov chains was introduced, a natural extension of continuous time Markov chains in which state holding times are determined by general continuous distributions.

A Semi-Markov chain can be considered as a Kripke structure in which the transitions are labelled by information about the speed at which the chain evolves from one state to another. In a Semi-Markov chain, the delay between two successive state changes can be generally distributed. This property has to be contrasted with continuous time Markov chains where these delays need to be governed by negative exponential distributions. In this section, we introduce the basic concepts of Semi-Markov chains.

A Semi-Markov chain M is a tuple (S, P, Q, L) with

$$AP \text{ the set of atomic propositions,}$$

$$S \text{ a finite set of states,}$$

$$P : S \times S \to [0, 1] \text{ the transition probability matrix}$$

$$Q : S \times S \text{ a matrix of continuous probability distribution functions and}$$

$$L : S \to 2^{AP} \text{ as the labeling function.}$$

The intuitive interpretation of a Semi-Markov chain is as follows. There exists a transition from state s to s' if and only if $P(s, s') > 0$. Matrix P determines the (discrete) probabilistic behaviour when changing from one state to another, i.e., $P(s, s')$ is the probability to move from state s to state s'. Note that this is identical to the probabilistic branching of a discrete time Markov chain; (S, P, L) is often called the embedded discrete time Markov chain of Semi-Markov chain M. Once a next state $s' \in S$ has been selected, the state holding time of state s is determined according to the probability distribution function $Q(s, s', t)$. Thus, $Q(s, s', t)$ denotes the probability to move from state s to s' within at most t time units, given that a transition from s to s' will be taken.

CSL can be used as a query language in the very same way as with continuous time Markov chains [21].

5.2.4 Discussion

The simplest way of accessing time information of course is to use discrete time Markov chains and discretize time into the different states. The problem with this simple approach is the resulting state space explosion, since for all states that already exist in the model we need to add states describing how much time it takes to go from the current state to the next state.

Since we can only operate model checking algorithms on finite state transition systems this means we would also have to define an upper limit of time. An example for discretizing time: Assume a transition system with 10^4 states. When adding discrete time with a discretization e.g. of 10 seconds and a maximum time of 24 hours this leads to 8640 additional states per already existing state describing the time spent in the current state before switching to the new state. In total this leads to $8.64*10^7$ states. When discretizing to a 1 second time this again extends the state space by a factor of 10. Also when increasing the maximum time up the state space can grow again. This is not a feasible way of adding time unless when discretizing very roughly, e.g. when timing information is different weekdays or similar.

Continuous time Markov chains allow for adding time into the transitions themselves via a rate parameter that describes the exact form of a negative exponential function. The stochastic processes described by continuous time Markov chains are characterised by the fact that the state holding times, indicating the amount of time the system stays in a state, are restricted to negative exponential distributions. As a result of their socalled memoryless property, the probability of moving from one state to another is independent of the amount of time the system has spent in the current state.

Although exponential distributions appropriately model a significant number of phenomena related to mass effects of random nature, in many occasions they are inadequate to faithfully model the stochastic behaviour of the system under consideration. This means that continuous time Markov chains should only be used for phenomena that can be modeled by negative exponential distributions. Otherwise predictions based on these models might be arbitrarily wrong.

Semi-Markov chains extend the idea of continuous Markov chains by allowing all possible

kinds of continuous probability distributions for the state holding times. This allows for a more precise results and for a broader application area since Semi-Markov chains are a generalization of continuous time Markov chains.

Chapter 6

Context prediction using Model Checking

In this chapter we will explain how we intend to perform Model Checking on Semi-Markov chains. Usually the system description is provided as a formula in temporal logics but with our approach we can directly build transition systems from user traces, thereby skipping the first step. We can then perform Model checking on those transition systems using our query operators.

6.1 Definition of Context Model

Context is a term with numerous meanings and shades of meaning. In order to provide a good definition we need to narrow its definition down to what is really essential. Since context computing as a research field is still young there are currently many different definitions of context. First of all, the term context is found quite regularly in common language use and is defined [22] in a generic way:

- the parts of a discourse that surround a word or passage and can throw light on its meaning
- the interrelated conditions in which something exists or occurs

[1] already has a definition of context referring to the interaction between devices and users, or more generally, the situations a device is used in:

- In archaeology, the context (physical location) of a discovery can be of major significance. See Stratification.
- In linguistics, context refers to the meaning of a word or phrase with regard to its place in the sentence and the topic being discussed.
- In computer science, context refers to the circumstances under which a device is being used, e.g. the current occupation of the user. (see also context awareness)

In [23] we find a very early definition of context. It clearly defines context to include more than merely geographical position information:

> "Three important aspects of context are: where you are, who you are with, and what resources are nearby [...]. Context encompasses more than just the users location, because other things of interest are also mobile and changing. Context includes lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation, e.g., whether you are with your manager or with a co-worker."

In general, we can identify at least the following aspects of context:

- geographical (e.g. country, street, building, floor, office)
- physical (e.g. lighting, noise level, temperature, acceleration, tilt)
- organizational (e.g. institution, department, group, project)
- social (e.g. family, friend, co-worker, married, single)
- emotional (e.g. heart rate, skin conductance)
- user (e.g. profile, location, capabilities, role, access rights)
- task (e.g. documenting, programming, building a house)
- action (e.g. typing, reading, walking, sitting, talking)
- technological (e.g. connectivity, network bandwidth, network latency)
- time (e.g. time of day, weekday, week, month, season of year)

Different applications in pervasive environments require different information being supplied in contexts. E.g. a tourist guide will need location data as context.

We will use the following definition of context found in [1] in our work:

> "Context is the surroundings, circumstances, environment, background, or settings which determine, specify, or clarify the meaning of an event."

6.1.1 Discrete context

Context is represented as a discrete property. Every state in our transition system reflects a different unique context. When referring to states in our transition systems we therefore always refer to a complete context representation. I.e. if we have the atomic propositions *location* and *activity* every state has to describe both atomic propositions. We may reason and query only one atomic proposition or any combination of atomic propositions we like, but states always describe all atomic propositions.

E.g. using the atomic propositions *location* and *activity*, possible states would be *location* = home $\wedge$ *activity* = watch tv or *location* = bus $\wedge$ *activity* = commute to work. A possible contexts could be *location* = bus meaning the activity does not matter, but it could also fully describe a situation like *location* = home $\wedge$ *activity* = watch tv.

6.2 Definition of State Transition System

Our transition system consists of states and transitions with probabilities and a discrete probability density function. Also there is an active state, a labeling function describing the atomic propositions that the state fulfills and the atomic propositions themselves.

6.2.1 Formal model

Our model for the state transition system is a modified Semi-Markov chain. We define our transition system M as tuple $M = (S, P, Q, L)$ with

$$S = \{s_1, s_2, s_3, \ldots, s_n\} \text{ is a finite set of states,}$$

$$P : S \times S \to [0, 1] \text{ with } \sum_{s' \in S} P(s, s') = 1$$

is the transition probability matrix for each transition $T_{s,s'}$ going from state s to state s',

$$Q : S \times S \to h_{s,s'}(x), \text{ with } h_{s,s'} : R_{\geq 0} \to [0, 1] \text{ and } Q(s, s') = h_{s,s'}(x)$$

the discrete probability density function matrix and a labelling function and

$$L : S \to 2^{AP}$$

which labels for each state which atomic propositions hold true for this state.

The continuous probability function matrix Q consists of a discrete probability density function $h_{s,s'}(t)$ for each pair of states $s, s' \in S$. Q describes the likelyhood of switching the context from state s to state s' at a time t.

The transition probability $P(s, s')$ computes as follows:

$$P(s, s') = \sum_{s' \in succ(s)} w_{s,s'}$$

6.2.2 Example scenario

A simple example would be the following scenario of an employee: After having breakfast he usually goes to work during week days, but sometimes, when sick, he decides to go back to bed. We are modeling this behaviour with a Semi-Markov chain. This translates into a transition system with three states and two transitions. We assume we already know some data from the employee's history in order to fill the transition system and the corresponding discrete probability density functions with some values.

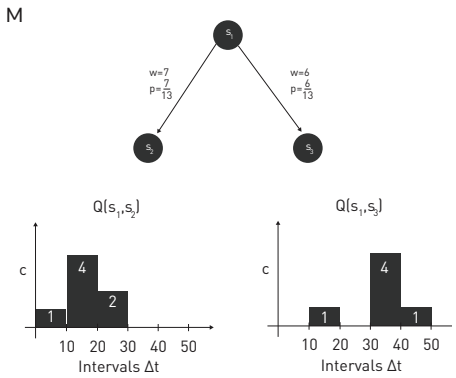

Figure 6.1: Example Semi-Markov chain M

When transforming these user traces into a Semi-Markov chain we might get a resulting transition system M that looks like figure 6.1. In this example we see three different states $S = \{s_1, s_2, s_3\}$ representing the different contexts that have been encountered when going through the user traces. These states are labeled with values for atomic propositions, in our example

$$L(s_1) = \text{activity=have breakfast} \wedge \text{location=dining room}$$

$$L(s_2) = \text{activity=go to work} \wedge \text{location=office}$$

$$L(s_3) = \text{activity=go back to bed} \wedge \text{location=bedroom}$$

Looking at figure 6.1 We immediately notice that the employee goes back to bed every day almost as often as he is going to work at the office. The transition weights w attached to the transitions display in numbers how often a context switch from a state occured to another state. In this example the transition weights are

$$w_{s_1,s_2} = 7, w_{s_1,s_3} = 6$$

The transition probability p display how often a transition is taken in comparison with all other transitions originating from the same state. In this example the transition probabilites are

$$P(s_1, s_2) = \frac{7}{13}, P(s_1, s_3) = \frac{6}{13}$$

Also displayed in figure 6.1 are the interval counts of the transitions' discrete probability density functions' $Q(s_1, s_2)$ and $Q(s_1, s_3)$. The interval size Δt in this example is set to 10 seconds. So if a context switch happens during the first 10 seconds it is added to the first interval in the corresponding discrete probability density functions, no matter if the context switch occured after 5 seconds or after 7 seconds.

$$c_{s_1,s_2}^{\lfloor \frac{t}{\Delta t} \rfloor} = \begin{cases} 1, \text{ if } t \in [0, 9] \\ 4, \text{ if } t \in [10, 19] \\ 2, \text{ if } t \in [20, 29] \\ 0, \text{ else} \end{cases}$$

The interval counts between state s_1 and state s_2 describe that a context switch with the context (s_1) = activity=have breakfast $\wedge$ location=dining room to a context $L(s_2)$ = activity=go to work $\wedge$ location=office happened one time during the first 9 seconds, four times between 10 and 19 seconds and two times between 20 and 29 seconds.

$$c_{s_1,s_3}^{\lfloor \frac{t}{\Delta t} \rfloor} = \begin{cases} 1, \text{ if } t \in [10, 19] \\ 4, \text{ if } t \in [30, 39] \\ 1, \text{ if } t \in [40, 49] \\ 0, \text{ else} \end{cases}$$

In the very same way the interval counts between states s_1 and state s_3 describe that a context switch with the context (s_1) = activity=have breakfast $\wedge$ location=dining room

to a context $L(s_3)$ = activity=go back to bed $\wedge$ location=bedroom happened one time during the first 9 seconds, four times between 30 and 39 seconds and one times between 40 and 49 seconds.

We can use this information to predict future contexts. We go into detail about how we compute these predictions in the next chapter.

6.3 Learning of probabilistic state transition system

One of the most important requirements for a context predictor is the ability to learn in a computationally efficient way. In this thesis we present a method to perform online learning on the fly.

When switching from one context to the next there are only two possible scenarios.

The first scenario is to that the next context maps to an already existing state in the transition system. Then again there are two different scenarios: Either there is already an edge from the current context to the next state, then we update the weight of that edge by increasing its weight by 1 and update the corresponding discrete probability density function, or there is no edge between the states yet. In that case we create an edge connecting the states with an initial weight of 1 and at the same time update its discrete probability density function.

The second scenario is that the new context is not yet represented in the transition system yet. In that case we create a new state state that incorporates the new context and create an edge from the current context to the new one with an initial weight of 1 and synchronously update its discrete probability density function.

In all cases the discrete probability density function that the edge represents has to reflect the time when contexts have been switched. This is achieved by taking the time difference of the time when the current context was entered and the time when the new context has been recognized.

Then we compute into which interval the context switch belongs to and update the interval count. Also the total count of hits is increased by one. The height of all intervals can

then be recomputed.

6.3.1 Learning of transition system

In order to transfer the data traces to Semi-Markov chains we read through the user trace data line by line. Each line represents a state in our transition system and the time when the state was entered. Each state is uniquely identified by the atomic propositions that hold true in the corresponding context. When we encounter a line we create a new state if the state is not already present in the transition system. If it is already present we do not need to create it since the states are uniquely defined by their atomic propositions assigned by the labelling function L. When encountering a new state also a transition between states must be added if it is not already present. If it is already present we do not add another transition.

6.3.2 Learning of transition probabilities

Upon coming up across a new context the transition system has to be adapted mirroring the new behaviour in order to allow for answering precisely to queries. In general there are two possible scenarios for being confronted by a new context:

1. the context has already been encountered before
2. the context never has been encountered before

In case 1 we again have two scenarios: Either we already have a transition from the current state to the new state, then we increase the weigth $w_{s,s'}$ by 1. In the other case of not already having a transition from the current state to the new state we initialize $w_{s,s'}$ with a value of 1.

In case 2 we have to add a new state to the transition system with the corresponding atomic propositions describing the context. Since the state is new there cannot be any transitions from any other state to this state yet, so we can set up the transition probability weight $w_{s,s'}$ with a value of 1.

Now we can recompute the total transition probability values on the fly

$$P(s, s') = \frac{w_{s,s'}}{\sum_{s'' \in succ(s)} w_{s,s''}}$$

6.3.3 Learning of transition times

Also the discrete probability density function has to be recomputed and reflect the time of the context switch.

$$I_i = [i \cdot \Delta t, (i+1) \cdot \Delta t)$$

A discrete probability density function in $Q(s, s')$ is divided into intervals I_i of a given interval size Δt which is provided as a system parameter. Each interval itself is a constant function which returns the relative probability of switching from context s to context s' in the given interval of time.

Each interval is associated with a relative probability via the mapping $h_{s,s'}(t)$

$$h_{s,s'} : \mathbb{R}_{\geq 0} \to [0, 1], h_{s,s'} : t \mapsto \frac{c_{s,s'}^{\lfloor \frac{t}{\Delta t} \rfloor}}{\Delta t \cdot w_{s,s'}},$$

with $c_{s,s'}^{\lfloor \frac{t}{\Delta t} \rfloor}$ being the count of context switches on transition $T_{s,s'}$ that occured in interval $I_{\lfloor \frac{t}{\Delta t} \rfloor}^{s,s'}$ and $w_{s,s'} = \sum_i c_{s,s'}^i$ being the total amount of context switches on transition $T_{s,s'}$. We divide by Δt in order for the integral $\int_0^\infty h_{s,s'}(t)dt$ to sum to 1.

So on a context switch along with the previously stated updates of transition weights and probabilities we also have to increase the transition's discrete probability density function by increasing the corresponding interval count by 1. Afterwards we need to recompute all heights for all intervals since the transition weight has changed.

6.3.4 Ageing

In order to adapt to changes in user behaviour over time we usually would assign every context update the same priority. When user behaviour changes over time though this can lead to bad quality of predictions. Especially when the behaviour before the change in user behaviour has been executed for a long time the weigth on e.g. the transitions are very high and don't allow for a quick rebalancing according to the new user behaviour. Since in chapter 4 we identified proactivity as a very important requirement we present an algorithm that allows for weighing of learning phases. We call this learning method ageing.

Suppose we have a number of user traces whose behaviour changes over time. I.e. picture a person who gradually changes his behaviour over time due to some external event that is not represented in the model. Then it would be good if we could choose how much emphasis we would like to put on the new data.

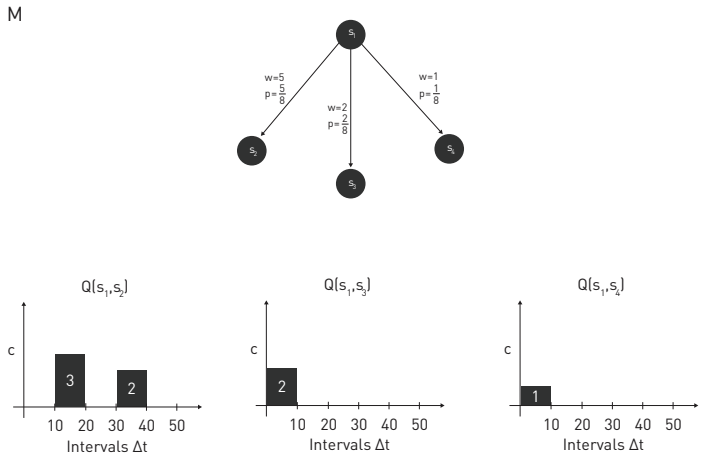

Figure 6.2: Ageing example Semi-Markov chain M

We can model this by separating the user traces into different periods each being represented by their own Semi-Markov chain. So let us assume two sequential periods of user behaviour modeled by Semi-Markov chains M and M'. Semi-Markov chains M represents the first period of user behaviour, Semi-Markov chains M' represents the modified user behaviour due to some external event.

We merge the behaviour into a new Semi-Markov chains M''. This is achieved by weighing the data from Semi-Markov chains M and M' respectively. In order to do this we use a linear parameter weight. We abbreviate the properties of a Semi-Markov chain by subscripting the corresponding name of the Semi-Markov chain. I.e. the probability matrix of Semi-Markov chain M' will be referred to as $P_{M'}(s, s')$.

In order to achieve linear weighing of the user traces in the two periods we use a parameter α. In order to put more emphasis on the current period we would naturally set α to a value smaller than 1.0. Setting α to 1.0 would disable ageing.

In order to use linear weighing of parameters x and y in order to obtain the result z we would compute the following formula:

$$z = \alpha \cdot x + (1 - \alpha) \cdot y,\ \alpha \in [0, 1]$$

First we need to recompute the transition weights ${w_{s,s'}}_{M''}$ for all transitions of Semi-Markov chains M and M'. Therefore we use the linear weight function for the recomputation of all transitions in the following way:

$${w_{s,s'}}_{M''} = \alpha \cdot {w_{s,s'}}_{M} + (1 - \alpha) \cdot {w_{s,s'}}_{M'}$$

When done computing all transition weights for Semi-Markov chain M'' we can now recompute the transition probabilities in Semi-Markov chain M''

$$P_{M''}(s, s') = \frac{{w_{s,s'}}_{M''}}{\sum_{s'' \in succ(s)} {w_{s,s''}}_{M''}}$$

We also recompute ${c_{s,s'}}_{M''}^{\lfloor \frac{t}{\Delta t} \rfloor}$ for the discrete probability density functions $Q_{M''}(s, s')$:

$${c_{s,s'}}_{M''}^{\lfloor \frac{t}{\Delta t} \rfloor} = \alpha \cdot {c_{s,s'}}_{M}^{\lfloor \frac{t}{\Delta t} \rfloor} + (1 - \alpha) \cdot {c_{s,s'}}_{M'}^{\lfloor \frac{t}{\Delta t} \rfloor} \text{for } t = k \cdot \Delta t, k \in \mathbb{N}$$

After computing the values for ${c_{s,s'}}_{M''}$ for all states s, s' we recompute the interval values for ${h_{s,s'}}_{M''}(x)$ using the interval size Δt and the also recomputed weights ${w_{s,s'}}_{M''}$. This leads to a new Semi-Markov chain M'' that has been generated through weighted learning out of the Semi-Markov chains M and M'.

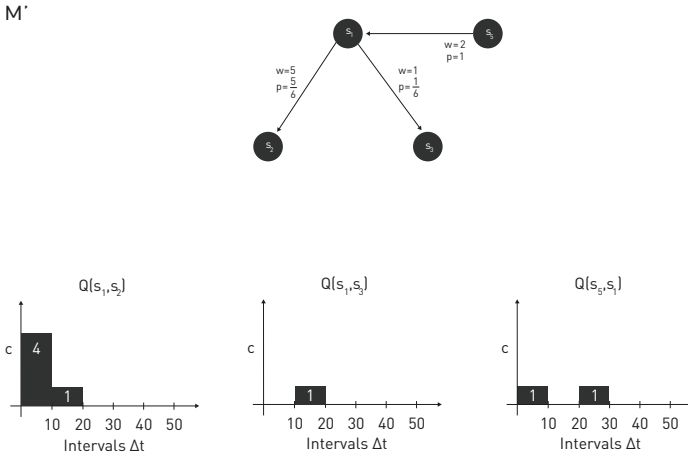

Figure 6.3: Ageing example Semi-Markov chain M'

With linear weighing we make sure that the discrete probability density functions of all transitions of M'' have the same properties as defined in the formal model, i.e. $\int_0^\infty h_{s,s'}(x)\,dx = 1$ and all other necessary properties like $\sum_{s' \in succ(s)} P(s, s')$.

We now demonstrate a simple example for our ageing algorithm. Given two Semi-Markov chains M and M' we want to create a new Semi-Markov chain M'' by using our ageing algorithm. Assuming we have already completed two phases of learning $PhaseP$ and $PhaseP'$ we will have two Semi-Markov chains M and M'. Now assuming that the user behaviour changed from learning phase P to learning phase P' we can combine these Semi-Markov chains with our ageing algorithm and therefore immediately react to incoming changes.

The transitions' interval counts in Semi-Markov chain M are

$$c_{s_1,s_2}^{\left\lfloor \frac{t}{\Delta t} \right\rfloor} = \begin{cases} 3, \text{ if } t \in [10, 19] \\ 2, \text{ if } t \in [30, 39] \\ 0, \text{ else} \end{cases}$$

The Semi-Markov chain M displayed in figure 6.2 displays a transition system with four states $S = \{s_1, s_2, s_3, s_4\}$. For simplicity's sake we decided to skip the labelling function $L(s)$ since they deliver more information about the contexts in the states, but we will not need this level of detail about states here.

Also displayed in figure 6.2 are the transitions weights

$$w_{s_1,s_2} = 5, w_{s_1,s_3} = 2, w_{s_1,s_4} = 1$$

and the transition probabilites.

$$P(s_1, s_2) = \frac{5}{8}, P(s_1, s_3) = \frac{2}{8}, P(s_1, s_4) = \frac{2}{8}$$

The other transitions' interval counts are defined likewise and we refer to figure 6.2 for details.

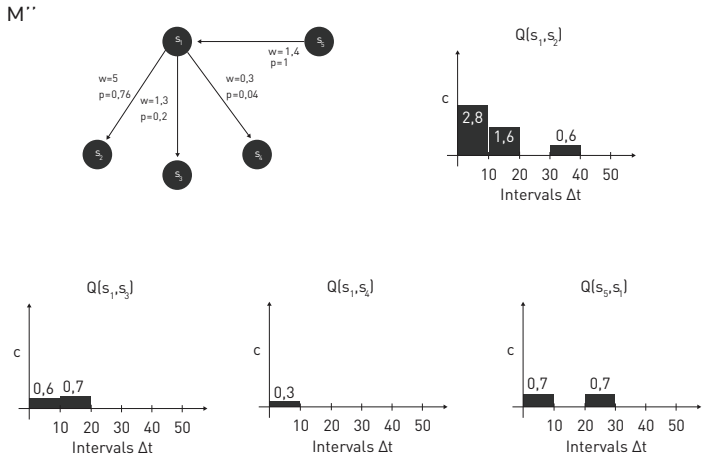

Figure 6.4: Semi-Markov chain M'', originating from applying ageing on Semi-Markov chains M and M'

The Semi-Markov chain M' displayed in figure 6.3 again displays a transition system with four states. The Semi-Markov chains M and M' have three states in common, but M' lacks state s_4 and instead has a state s_5. Also different are the transition weights and probabilities as well as the interval counts for the discrete probability density functions $Q(s, s')$.

Figure 6.4 displays Semi-Markov chain M'', the result of the merging of M and M. As you can see M'' has all states that M and M' have as well as all their transitions. The transition weights were recomputed with the formula given above. Also the transition probabilities were recomputed using the already recomputed transition probabilities. As

you can ageing was also applied on the interval counts, leading to real numbers instead of natural numbers for the interval counts. For this example we chose the ageing parameter $\alpha = 0.3$, giving Semi-Markov chain M' a weight of 70% and Semi-Markov chain M a weight of 30%. Assuming Semi-Markov chain M' is a transition system from a younger phase than M this would put an emphasis on sudden changes in user behaviour.

6.3.5 Incorporation of uncertainty in states

This can be achieved by extending the active state by using a probability vector of all states with probabilities for being in each state.

$$\Omega = \{s_1, s_2, s_3, \cdots, s_n\}$$

$$S : \omega^n \rightarrow [0,1]^n, \omega \in \Omega$$

So the vector S now is a probability distribution over all states, assigning a probability to each state to be the current state. The values of the vector have to sum up to less or equal 1, i.e. $\sum_{i=1}^{n} S_i \leq 1.0$. S_k would then return the probability of being in state s_k at the current time.

When updating the transition system with a new context the new context supplied from the sensors would also have to be labelled with a uncertainty value $v \in [0,1]$.

6.4 Definition of query interface

Here we present our interface for registering queries.

6.4.1 Active queries

Often times an application will need to have information like based on the current time a device has spent in the current context, what is the next most likely context. We call these types of queries "active queries" or "one shot queries". So basically using these queries an application can pull information from the context prediction framework. The context predictor immediately computes an active query upon receiving it and returns its value given the current context.

6.4.2 Continuous queries

In contrast to active queries continuous queries describe a different query type: continuous queries allow for a more "push-like" notifying than the "pull-like" active queries.

This is very important since certain operations need to perform some kind of polling in order to work correctly. But using active queries in order to poll the context prediction framework will result in the repeated computation of one and the same query which may not be feasible on devices with very low computational power, small batteries and very limited memory. Also the timely delivery of a notification could be affected since polling can only be done in intervals.

They will be reevaluated continuously after a certain time period has passed and on every context switch. The result of the query will be delivered to the querying application via a notification. On registering the query the application has to state the lifetime of the query and when it would like to get a notification, i.e. on every time the query becomes true, only for the first k times, or for a time window of t seconds.

The life time of passive queries needs to be restricted. We propose different mechanisms for this depending on the number of positive evaluations, time and number of context

switches that have already happened. This way the application designers can specifiy the point from which on a passive query can be ignored by the context predictor.

6.5 Query specifications and semantics

Unlike PCTL we do not only employ the Next and the Until operators but also several operators which are similar in use but offer a wider range of expressiveness.

6.5.1 $X_{P \cong p}(\varphi)$

This operator will return *true*, if the probability of the next active context $\tau \models \varphi$ is greater than a probability threshold parameter p. So the arguments for the operator are the probability threshold and a context φ. Otherwise, if the probability is less than the probability threshold or if there is no successor state modeling *varphi* the result will be $false$. This operator does not take time into account.

Example query The following query specifies a search context *activity* that should equal the value *sleep* and a probability threshold p of 0.2. If there is the possibility of entering a context that specifies the atomic proposition *activity* to be *sleep* within one transition step with a probability of less than 0.2 the result will be *true*. If the probability is higher than 0.2 the result will be $false$.

$$X_{P \leq 0.2}(activity = sleep)$$

6.5.2 $\hat{X}$

This operator returns the most likely next context to happen after the current one. It does not take time into account. If a state does not have any successor states the result is empty.

Example query This query does not have any arguments and specifies a most likely next state query and will return the most likely context that can be reached within one transition step in the transition system.

6.5.3 $X_{P\cong p}^{T\geq t}(\varphi)$

This operator will return *true*, if the probability of the next active context $\tau \models \varphi$ is greater than a probability threshold parameter p after having already spent t seconds in the current state. Otherwise, if the successor states do not model φ the result will be $false$.

Example query The following query specifies a search context *location* that should equal the value *home*, the time spent in the current state of 20 seconds and a minimum probability of 0.7 If the collected data suggests that there is a probability of greater than 0.7 to reach a the next context with the properties of $location = home$ with the next transition step given the time spent in the current context being 20 seconds the result will be *true*. Otherwise, if the likelihood is less than 0.7 to reach the next context with the property $location = home$ within the next transition step the result will be $false$.

$$X_{P\geq 0.7}^{T\geq 20}(location = home)$$

6.5.4 $\hat{X}^{T\geq t}$

This operator returns the most likely next context to happen after the current context after having spent t seconds in the current state. If a state does not have any successor states the result is empty.

Example query The following query specifies the time already spent in the current state of 20 seconds and will return the most likely state to follow up the current context, given that 60 seconds have already passed in the current context.

$$\hat{X}^{T \geq 60}$$

6.5.5 $X_{P \cong p}^{t \leq T \leq t+\Delta t}(\varphi)$

This operator returns *true* if the probability of the next active context $\tau \models \varphi$ is greater than a probability threshold parameter p after having spent t seconds in the current state within the next Δt seconds. If there are not successor states that model φ or if the corresponding context switches occur after Δt seconds, the result will be $false$.

Example query The following query specifies a search context *location* that should equal the value *home* and *activity* that should equal *sleep*, the time spent in the current state of 60 seconds, a time window of 9 minutes and a minimum probability of 0.8 If the collected data suggests that there is a probability of greater than 0.8 to reach a context with the next transition that suggests the user sleeps at home, given the time spent in the current context being 60 seconds the result will be *true*. Otherwise, if the likelihood is less than 0.8 to reach the next context with the property $location = home \wedge activity = sleep$ within the next transition step in the next 9 minutes the result will be $false$.

$$X_{P \geq 0.8}^{60 \leq T \leq 600}(activity = sleep \wedge location = home)$$

6.5.6 $\hat{X}^{t \leq T \leq t+\Delta t}$

This operator returns the most likely next context to happen after the current context after having spent t seconds in the current state, within the next Δt seconds. If a state does not have any successor states the result is empty.

Example query The following query specifies will result in the most likely context that can be reached within one transition, given that 1800 seconds have already passed in the current context and assuming this context switch is going to happen within the next 30 minutes.

$$\hat{X}^{1800 \leq T \leq 3600}$$

6.5.7 $\Phi U_{P \cong p}^{t \leq T \leq t+\Delta t} \Psi$

Given the currently already spent time in the active state t this operator returns *true* if there is at least one path from the current context that models the following properties within the next Δt seconds. There has to be a future context modeling Φ and all contexts before this context on this path have to model Ψ, starting from the state after the current active state. If at least one such path exists the result will be *true*, otherwise if there is no such path at all or not within the specified time the result will be $false$.

Example query The arguments for the following query are the time already spent in the current context, 30s, the time window we are interested in, 60s, a probability threshold p of 0.3 and an atomic proposition *location* with the value office. This query will result in *true*, if there exists at least one path starting from the current state that has the atomic proposition *location* model $office$ within 60s and its probability is higher than 0.3. If that is not the case the result will be $false$.

$$\text{true } U_{P \cong 0.3}^{30 \leq T \leq 90} \text{ location} = \text{office}$$

6.5.8 $X^{-1}_{P \cong p}(\varphi)$

Given a context φ for the next context and a probability boundary p this operator returns the time $t_{predicted}$, if there is a context following up the current context that models φ and the probability of switching to this new context is higher than p. If there is no such next context modeling φ or if it is not possible to reach it with a probability of higher than p the result will be empty.

Example query Given a probability threshold p of 0.9 and an atomic proposition $activity = swim$ this operator will return the time it takes for the next context switch to a context in which the user is swimming with a probability of higher than 0.9 that is reachable within one transition step. If there is such a context reachable, the result will be the time $t_{predicted}$ it takes to enter that state. Otherwise the result will be empty.

$$X^{-1}_{P>0.9}(activity = swim)$$

6.6 Computation of context predictions

6.6.1 Model Checking algorithms for Semi-Markov Processes

This section deals with the mathematical foundation of computation of queries.

We define the multivariate random variable X with $X_i \in S$ and X_i, describing the active state at step i in path σ. I.e. $X_k = s_k$ designates that in path σ after k context switches the active state is state s_k. We also define the multivariate random variable S with $S_i \in \mathbb{R}$, describing the time that has already passed in the current state at step i.

We can also reason about certain path properties for a path σ. $\sigma@t$ defines the atomic propositions $L(s)$ for the state that is active in this path at time t. We also assume a path always starts at a state with time $t = 0$. $\sigma[i]$ describes the atomic propositions $L(s)$ valid in state s for s being the ith state following the start state in path σ. We naturally assume that the starting state of path σ has the index $i = 0$.

In the following subsections we will go into detail about how to compute the results for the different operators. Since the most likely next operators rely on the computation of the next operators we will explain how we computed and evaluated the most likely next operators in the evaluation section 7.5.

6.6.2 $X_{P \cong p}(\varphi)$

Given a probability threshold P and an atomic proposition property φ this operator will return *true* if the probability of switching from the current context to a context that models φ is fulfills $P \cong p$. For example, with $P \geq p$ the probability would have to be greater or equal than the probability parameter P. If there is no such context reachable within one step in the transition system or if the probability p is not fulfilling $P \cong p$ the operator will return $false$. The operator does not take time into account but only the transition probabilities.

So given the random variable X with $X_i = s$, the current state in path σ and $X_{i+1} = s'$, designating the next state in path σ with $s' \models \varphi$ we can argue

$$Pr\left(X\left(\varphi\right)\right) = \sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s' \,|X_i = s\right) = \sum_{s' \in succ(s), s' \models \varphi} P\left(s, s'\right)$$

I.e. the probability of switching to a state witch fulfills the property φ given the current state is state s, is the sum of all transition probabilities between states s and direct successor states s' that fulfill φ. The operator will then return $true$ or $false$, depending on the relation chosen and the probability threshold.

6.6.3 $X^{T \geq t}_{P \cong p}\left(\varphi\right)$

Given a probability threshold P, an atomic proposition property φ and the current time already spent in the current state t this operator will return $true$ if the probability of switching from the current context to a context that models φ is fulfills $P \cong p$, given that time t has already passed. For example, with $P \geq p$ the probability would have to be greater or equal than the probability parameter P. If there is no such context reachable within one step in the transition system given the current time t or if the probability p is not fulfilling $P \cong p$ the operator will return $false$. Basically the operator returns the "rest probability" to switch to a context that fulfills φ after spending time t in state s.

So given the random variable X with $X_i = s$, the current state in path σ and $X_{i+1} = s'$, designating the next state in path σ with $s' \models \varphi$ and the time already spent in the current state $S_i = t$ we can compute the probability of switching contexts by

$$Pr\left(X^{T \geq t}\left(\varphi\right)\right) = \sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s' \,|X_i = s, S_i > t\right)$$

I.e. the probability of going to state s' that fulfills φ, given the current state s and the time that has already passed t. Using the Bayesian Rule we can compute

$$= \frac{\sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s', S_i > t, X_i = s\right)}{P\left(S_i > t, X_i = s\right)}$$

Marginalizing the denominator we get

$$= \frac{\sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s', S_i > t, X_i = s\right)}{\sum_{s'' \in succ(s)} P\left(X_{i+1} = s'', S_i > t, X_i = s\right)}$$

Using the Bayesian Rule again we get

$$= \frac{\sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s', S_i > t \,|X_i = s\right) \cdot P\left(X_i = s\right)}{\sum_{s'' \in succ(s)} P\left(X_{i+1} = s'', S_i > t \,|X_i = s\right) \cdot P\left(X_i = s\right)}$$

Now canceling out $P\left(X_i = s\right)$ we get

$$= \frac{\sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s', S_i > t \,|X_i = s\right)}{\sum_{s'' \in succ(s)} P\left(X_{i+1} = s'', S_i > t \,|X_i = s\right)}$$

with

$$P\left(X_{i+1} = s', S_i > t \,|X_i = s\right) = P\left(s, s'\right) \cdot \delta_{s,s'}\left(t\right)$$

and

$$\delta_{s,s'}\left(t\right) = \begin{cases} 1, \text{ if } t > t_{max}\left(h_{s,s'}\left(t\right)\right) \text{ and } \left\{s' \in succ\left(s\right) : s' \models \varphi, |succ\left(s\right)| = 1\right\} \\ H^{\geq}_{s,s'}\left(t\right), \text{ else} \end{cases}$$

The operator $\delta_{s,s'}\left(t\right)$ ensures that given only one successor state s' the operator will return 1 even though the maximum time recorded in $h_{s,s'}\left(t\right)$ is lower than the time t already spent in state s. If there is no successor state for state s the result will be 0.

The computation of factor time follows here

$$H^{\geq}_{s,s'}\left(t\right) = \int_t^{\infty} h_{s,s'}\left(x\right) dx$$

Therefore

$$Pr\left(X^{T\geq t}\left(\varphi\right)\right) = \frac{\sum\limits_{s'\in succ(s), s'\models\varphi} P\left(s,s'\right)\cdot\delta_{s,s'}\left(t\right)}{\sum\limits_{s''\in succ(s)} P\left(s,s''\right)\cdot\delta_{s,s''}\left(t\right)}$$

In words we sum up the product of transition probabilities to sucessor states that fulfill *varphi* with the integral over the discrete probability density function of the corresponding transition starting from time t and divide this by doing the same for all successor states. If the resulting value fulfills $P \cong p$ the result of the operator is $true$, otherwise the result is $false$.

6.6.4 $X_{P\cong p}^{t\leq T\leq t+\Delta t}\left(\varphi\right)$

Given a probability threshold P, an atomic proposition property φ, the current time already spent in the current state t and a time window of size Δt this operator will return $true$ if the probability of switching from the current context to a context that models φ is fulfills $P \cong p$, given that time T has already passed and the context switch will occur during the time window in the time interval $[t, t+\Delta t]$. If there is no such context reachable within one step in the transition system given the current time T and the time window or if the probability p is not fulfilling $P \cong p$ the operator will return $false$. Basically the operator returns the "rest probability" to switch to a context within the time window that fulfills φ after spending time t in state s.

So given the random variable X with $X_i = s$, the current state in path σ and $X_{i+1} = s'$, designating the next state in path σ with $s' \models \varphi$ and the time already spent in the current state $t \leq S_i \leq t + \Delta t$ we can compute the probability of switching contexts by

$$Pr\left(X^{t\leq T\leq t+\Delta t}\left(\varphi\right)\right) = \sum_{s'\in succ(s), s'\models\varphi} P\left(X_{i+1} = s', S_i \leq t+\Delta t \,|X_i = s, S_i > t\right)$$

I.e. the probability of going to state s' that fulfills φ with spending less than time $t + \Delta t$ in the current state, given the current state s and the time that has already passed t.

Using the Bayesian Rule we can compute

$$= \frac{\sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s', S_i > t, S_i \leq t + \Delta t, X_i = s\right)}{P\left(S_i > t, X_i = s\right)}$$

Marginalizing the denominator we get

$$= \frac{\sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s', S_i > t, S_i \leq t + \Delta t, X_i = s\right)}{\sum_{s'' \in succ(s)} P\left(X_{i+1} = s'', S_i > t, X_i = s\right)}$$

Using the Bayesian Rule again we get

$$= \frac{\sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s', S_i > t, S_i \leq t + \Delta t \,|X_i = s\right) \cdot P\left(X_i = s\right)}{\sum_{s'' \in succ(s)} P\left(X_{i+1} = s'', S_i > t \,|X_i = s\right) \cdot P\left(X_i = s\right)}$$

Now canceling out $P\left(X_i = s\right)$ we get

$$= \frac{\sum_{s' \in succ(s), s' \models \varphi} P\left(X_{i+1} = s', S_i > t, S_i \leq t + \Delta t \,|X_i = s\right)}{\sum_{s'' \in succ(s)} P\left(X_{i+1} = s'', S_i > t \,|X_i = s\right)}$$

with

$$P\left(X_{i+1} = s', S_i > t, S_i \leq t + \Delta t \,|X_i = s\right) = P\left(s, s'\right) \cdot \left(H^{\geq}_{s,s'}\left(t\right) - H^{\geq}_{s,s'}\left(t + \Delta t\right)\right)$$

In words, this is the transition probability between states s and s' multiplied with the integral $\int_t^{t+\Delta t} h_{s,s'}\left(x\right) dx = H^{\geq}_{s,s'}\left(t\right) - H^{\geq}_{s,s'}\left(t + \Delta t\right)$

$H^{\geq}_{s,s'}\left(t\right)$ is again defined as

$$H^{\geq}_{s,s'}\left(t\right) = \int_t^{\infty} h_{s,s'}\left(x\right) dx$$

Now with the denominator

$$P\left(X_{i+1} = s', S_i > t \,|X_i = s\right) = P\left(s, s'\right) \cdot \delta_{s,s'}\left(t\right)$$

again with

$$\delta_{s,s'}(t) = \begin{cases} 1, \text{ if } t > t_{max}(h_{s,s'}(t)) \text{ and } \{s' \in succ(s) : s' \models \varphi, |succ(s)| = 1\} \\ H^{\geq}_{s,s'}(t), \text{ else} \end{cases}$$

The operator $\delta_{s,s'}(t)$ again ensures that given only one successor state s' the operator will return 1 even though the maximum time recorded in $h_{s,s'}(t)$ is lower than the time t already spent in state s. If there is no successor state for state s the result will be 0.

Therefore

$$Pr\left(X^{T\leq t\leq T+\Delta T}(\varphi)\right) = \frac{\sum_{s'\in succ(s), s'\models\varphi} P(s,s') \cdot \left(H^{\geq}_{s,s'}(t) - H^{\geq}_{s,s'}(t+\Delta t)\right)}{\sum_{s''\in succ(s)} P(s,s') \cdot \delta_{s,s'}(t)}$$

In words this means that the probability of a context switch happening to a state fulfilling φ before time $t + \Delta t$ given that the already spent time in the current state is t and the current state being s is the sum of the products of the transition probabilities between state s and its successor states s' that model φ and the integral of the corresponding discrete probability density functions in the specified time interval divided by the sum of the products of transition probabilities to sucessor states that fulfill *varphi* with the integral over the discrete probability density function of the corresponding transition starting from time t.

6.6.5 $\Phi U^{t\leq T\leq t}_{P\cong p}\Psi$

Given two atomic proposition formulas Φ and Ψ, the time spent in the current state t, a time in the future $t + \Delta t$ and a probability threshold p this operator will return true, if there exists a path starting in the current state, with all states fulfilling Φ until a state is reached that fulfills Ψ in a time window $[t, t + \Delta t]$ and also fulfills $P \cong p$. If there is no such path in the specified time window the result will be $false$.

Before we start computing $Pr(\Phi U^{t\leq T\leq t}\Psi)$ we first reintroduce some definitions.

A path in a Semi-Markov chain $M = (S, R, L)$ is defined as a sequence of states and transitions $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} s_3 \xrightarrow{t_3} \ldots s_k \xrightarrow{t_k}$ with $s_i \in S$, $t_i \in \mathbb{R}_{\geq 0}$ and $R(s_i, s_{i+1}) > 0$.

We can then define $Path\,(s)$ as the set of of all paths starting in state s:

$$Pr^{T\cong t}\,(s,\varphi) = \{\sigma \in Path\,(s)\,|\sigma \models \varphi\}$$

defines the probability measure for all paths σ starting in state s and fulfilling a path formula φ, making the relation $t \cong T$ true. So for example $Pr^{T\leq 30}\,(s_5, location = bathroom)$ defines the probability measure for all paths σ that start in state $s5$ and that take place in the location bathroom within the next 30 seconds.

Now we can reason about the operator:

$$Pr(\Phi U^{t\leq T\leq t}\Psi) = \sum_{\sigma\in Path(s)} P\,(\sigma\,[k] = \Phi, \sigma\,[j] = \Psi, S_{i+k} \leq t + \Delta t\,|X_i = s, S_i > t\,)\ \forall 0 \leq j < k$$

So, given the current state s and the time that has already passed in this state t we want to infer the probability of reaching a state within time Δt that fulfills Φ in some path σ at a transition step k with this path also fulfilling Ψ for all earlier transition steps.

Using the Bayesian Rule we can compute

$$= \frac{\sum_{\sigma\in Path(s)} P\,(\sigma\,[k] = \Phi, \sigma\,[j] = \Psi, S_{i+k} \leq t + \Delta t, X_i = s, S_i > t)}{P\,(X_i = s, S_i > t)}\ \forall 0 \leq j < k$$

Marginalizing the denominator we get

$$= \frac{\sum_{\sigma\in Path(s)} P\,(\sigma\,[k] = \Phi, \sigma\,[j] = \Psi, S_{i+k} \leq t + \Delta t, X_i = s, S_i > t)}{\sum_{\sigma\in Path(s)} P\,(S_{i+k} \leq t + \Delta t, X_i = s, S_i > t)}\ \forall 0 \leq j < k$$

Using the Bayesian Rule again we get

$$= \frac{\sum_{\sigma\in Path(s)} P\,(\sigma\,[k] = \Phi, \sigma\,[j] = \Psi, S_{i+k} \leq t + \Delta t, S_i > t\,|X_i = s\,) \cdot P\,(X_i = s)}{\sum_{\sigma\in Path(s)} P\,(S_{i+k} \leq t + \Delta t, S_i > t\,|X_i = s\,) \cdot P\,(X_i = s)}\ \forall 0 \leq j < k$$

Now canceling out $P(X_i = s)$ we get

$$= \frac{\sum_{\sigma \in Path(s)} P(\sigma[k] = \Phi, \sigma[j] = \Psi, S_{i+k} \leq t + \Delta t, S_i > t \,|X_i = s)}{\sum_{\sigma \in Path(s)} P(S_{i+k} \leq t + \Delta t, S_i > t \,|X_i = s)} \; \forall 0 \leq j < k$$

In words the nominator describes the sum of all paths starting in state s that fulfill $\Phi U \Psi$ within the time interval $[t, t + \Delta t]$ whereas the denominator describes the sum of all paths starting in state s that can be reached within the time interval $[t, t + \Delta t]$.

Using our probability measure definition from above we get

$$\sum_{\sigma \in Path(s)} P(\sigma[k] = \Phi, \sigma[j] = \Psi, S_{i+k} \leq t + \Delta t, S_i > t \,|X_i = s)$$
$$= Pr^{t \leq T \leq t+\Delta t}(s, \varphi) \; \forall 0 \leq j < k$$

for

$$\varphi = \{\sigma \,|\exists x \in [t, t + \Delta t] : \sigma@x \models \Psi \wedge \forall y \in [t, x] : \sigma@y \models \Phi \wedge \sigma[k] = \Phi, \sigma[j] = \Psi\} \; \forall 0 \leq j < k$$

and

$$\sum_{\sigma \in Path(s)} P(S_{i+k} \leq t + \Delta t, S_i > t \,|X_i = s) = Pr^{t \leq T \leq t+\Delta t}(s, true)$$

for nominator and denominator.

The probability of taking a path computes by chaining transition probability density functions $f_A, f_B, \cdots f_K$ to each other with the $\circ$ operator according to [24]:

$$f_X(x) \circ f_Y(y) = f_Z(z_i) = \sum_{x \cdot y = z_i} f_X(x) \cdot f_Y(y)$$

I.e. the discrete probability density function for one path σ from state s_1 to state s_k over states $s_i, 2 \leq i < k$ would be $\overset{\circ}{H}_\sigma(x) = h_{s_{k-1},s_k}(x) \circ \ldots \circ h_{s_2,s_3}(x) \circ h_{s_1,s_2}(x)$. Using this information we can now compute the probability measure for paths

$$Pr^{t \leq T \leq t+\Delta t}(s, \varphi) = \sum_{\sigma \in Path(s) | \sigma \models \varphi} \int_t^{t+\Delta t} \overset{\circ}{H}_\sigma(x)\, dx = \int_t^{t+\Delta t} \sum_{\sigma \in Path(s) | \sigma \models \varphi} \overset{\circ}{H}_\sigma(x)\, dx$$

Input: The probability threshold p, the current state s, atomic propositions φ
Output: The time t at which probability p is reached
declare Stack stack
declare DPDF nominator, denominator
nominator.weight $\leftarrow 1$
denominator.weight $\leftarrow 1$
for all States $s' : P(s, s') > 0$ **do**
    stack.push(new TreeSearchNode($s', h_{s,s'}$, $P(s, s')$, $t_{min}(h_{s,s'})$))
end for
while $\neg$ stack.isEmpty() **do**
    declare TreeSearchNode tsn $\leftarrow$ stack.pop()
    declare TreeSearchNode tmp
    declare State $s_k \leftarrow$ tsn.state()
    if tsn.getTime() $\leq t \wedge$ tsn.$P(s, s')$ > min. probability **then**
        if $s_k \models \varphi$ **then**
            nominator $\leftarrow$ nominator + tsn.$P(s, s') \cdot$ tsn.$h_{s,s'}$
        end if
        for all States $s' : P(s_k, s') > 0$ **do**
            if tsn.time() + $t_{min}(h_{s,s'}) \leq t$ **then**
                tmp $\leftarrow$ new TreeSearchNode($s', h_{s,s'} \circ$ tsn.$h_{s,s'}, P(s_k, s') \cdot$
                tsn.$P(s, s')$, tsn.time() + $t_{min}(h_{s,s'})$)
                stack.push(tmp)
            else
                denominator $\leftarrow$ denominator + $P(s_k, s') \cdot h(s_k, s')$
            end if
        end for
    end if
end while
return $\dfrac{\int_0^{t+\Delta t} \text{nominator}(x)\,dx}{\int_0^{t+\Delta t} \text{denominator}(x)\,dx}$

Algorithm 1: The algorithm for computing $\Phi U^{t \leq T \leq t} \Psi$

Eventually we get

$$Pr(\Phi U^{t\leq T\leq t}\Psi) = \frac{Pr^{t\leq T\leq t+\Delta t}(s,\varphi)}{Pr^{t\leq T\leq t+\Delta t}(s,true)} = \frac{\int_t^{t+\Delta t} \sum_{\sigma\in Path(s)|\sigma\models\varphi} \overset{\circ}{H}_\sigma(x)\,dx}{\int_t^{t+\Delta t} \sum_{\sigma\in Path(s)|\sigma\models true} \overset{\circ}{H}_\sigma(x)\,dx}$$

for

$$\varphi = \{\sigma\,|\exists x\in[t,t+\Delta t]:\sigma@x\models\Psi\wedge\forall y\in[t,x]:\sigma@y\models\Phi\wedge\sigma[k]=\Phi,\sigma[j]=\Psi\}\ \forall 0\leq j<k$$

In order to find all possible paths originating from state s we perform a depth-first-search over the transition system starting from the active state s and stopping when time $t+\Delta t$ is reached or when the probability on the path has decreased to a value lower than a probability threshold that is provided as a system parameter.

During the search we chain the probabilities for all paths and add them together. We keep the results for the nominator and the denominator separate.

The total probability measure for all paths $\sigma\in Path(s)\models\exists x\in[0,t]:\sigma@x\models\Psi\wedge\forall y\in[0,x]:\sigma@y\models\Phi$ is the sum of all discrete probability density functions that we obtained by chaining their respective discrete probability density functions. In the very same way we can sum up all discrete probability density functions for all paths $\sigma\in Path(s)\models\exists x\in[0,t]:\sigma@x\models\Psi\wedge\forall y\in[0,x]:\sigma@y\models true$.

The next step is to integrate on the results for the newly created discrete probability density function in the nominator and the denominator from time t to time $t+\Delta t$, returning the probability we were looking for. The operator results in $true$, if the result fulfills $P\cong p$, otherwise in $false$.

6.6.6 $X^{-1}(p,\varphi)$

This operator returns the most likely time it takes to make a context switch from the current state to a state fulfilling φ for a probability of at least p.

In order to do this we need to define the inverse function of $X^{T\leq t}(\varphi)$:

$$Pr\left(X^{T\leq t}(\varphi)\right) = \sum_{s'\in succ(s), s'\models\varphi} P\left(S_i \leq t, X_{i+1} = s' | X_i = s\right)$$

$$= \sum_{s'\in succ(s), s'\models\varphi} P(s,s') \cdot H^{\leq}_{s,s'}(t)$$

$$= \sum_{s'\in succ(s), s'\models\varphi} P(s,s') \cdot \int_0^t h_{s,s'}(x)\,dx$$

$$= \int_0^t \sum_{s'\in succ(s), s'\models\varphi} P(s,s') \cdot h_{s,s'}(x)dx$$

$$= \int_0^t \hat{h}(x)\,dx = \hat{H}^{\leq}(t)$$

The function $\hat{h}(x)$ is defined as the product $\sum_{s'\in succ(s), s'\models\varphi} P(s,s') \cdot h_{s,s'}(x)$, whereby the sum f_z of two CDFs f_x and f_y is defined by summing the values P_i for each corresponding interval I_i as described in [24].

$$P_i^Z = P_i^X + P_i^Y$$

In the very same way we define $\hat{H}^{\leq}(t)$ to be the integral $\int_0^t \hat{h}(x)\,dx$.

Now we get the inverse function by

$$\hat{H}^{\leq^{-1}}\left(Pr\left(X^{T\leq t}(\varphi)\right)\right) = \hat{H}^{\leq^{-1}}\left(\hat{H}^{\leq}(t)\right) = t, \quad s' \in succ(s), s' \models \varphi$$

So given a probability p as an input, $\hat{H}^{\leq^{-1}}_{s,s'}(p)$ will give the time $t_{fire} = t$ as an output when the context switch will occur with the given probability p.

In general for discrete probability density functions we cannot easily compute $t_{fire} = \hat{H}^{\leq^{-1}}_{s,s'}(p)$, but in our discrete model of a discrete probability density function it is fairly easily possible.

To compute $t_{fire} = \hat{H}^{\leq^{-1}}_{s,s'}(p)$ we simply do a step by step integration over the discrete probability density function $\hat{h}_{s,s'}(t)$ for each interval, beginning with the first interval. As soon as the value of the integral is bigger than p, we undo the last addition. Now we

Input: The probability threshold p, the current state s, atomic propositions φ
Output: The time t at which probability p is reached

declare $\hat{h}(x)$
declare $time = 0$, $sum = 0$

for all States $s' : P(s, s') > 0 \wedge s' \models \varphi$ **do**
    for all $0 \leq t \leq t_{max}(h_{s,s'}(x))$ **do**
        $\hat{h}(i) \leftarrow \hat{h}(i) + P(s, s') \cdot h_{s,s'}(x)$
    end for
end for

for all $0 \leq t \leq t_{max}\left(\hat{h}(x)\right)$ **do**
    if $c^t_{\hat{h}(x)} > 0$ **then**
        if $sum + c^t_{\hat{h}(x)} \cdot \Delta t < p$ **then**
            $sum \leftarrow sum + c^t_{\hat{h}(x)} \cdot \Delta t$
        else
            return $time + ((p - sum)/\hat{h}(t))$
        end if
        $time \leftarrow time + \Delta t$
    end if
end for

return false

Algorithm 2: The algorithm for finding t_{fire}

can sum up the number of intervals and multiply them by the given interval size and add $\frac{p-sum}{\hat{h}_{s,s'}(lastinterval)}$, this returns exactly the time $t_{fire} = t$ we were looking for.

$$t_{fire} = \underbrace{\Delta t \cdot (k-1)}_{\text{sum of interval times except for the last interval}} + \underbrace{\frac{p - sum}{\hat{h}\,(k \cdot \Delta t)}}_{\text{surface of the last interval}}$$

with the interval sizes Δt provided as a system parameter.

with a k such that

$$\sum_{i=0}^{k-1} \hat{h}\,(i \cdot \Delta t) \leq p < \sum_{i=0}^{k} \hat{h}\,(i \cdot \Delta t)$$

The surface of the last interval computes by the following equations. Since $\hat{h}\,(t)$ is a constant function in each interval, when integrating we therefore get a linear function. So when writing $\hat{H}\,(t)$ we have a linear function $y = mx + c$ with c being the sum of all surfaces of the earlier intervals of $\hat{h}\,(x)$. We can compute $m = \frac{\Delta y}{\Delta x} = \frac{\Delta t \cdot \hat{h}(i) - 0}{\Delta t}$ and get

$$y = \hat{h}\,(i) \cdot x + sum$$
$$\iff x = \frac{y - sum}{\hat{h}\,(i)}$$

Since y designates the accumulated probability and x displays the time we get

$$t = \frac{p - sum}{\hat{h}\,(i)}$$

We illustrate the algorithm for finding t_{fire}

6.6.7 Prediction of formula satisfaction time

The prediction of formula satisfaction time can be computed using $X^{-1}\,(p, \varphi)$.

6.6.8 Incorporation of uncertainty in states

It is not always one hundred percent clear in which context we actually are because context recognition is almost always based on sensor data which can be blurred by random noise.

In order to incorporate uncertainty into the results for queries like $X(\varphi)$ etc. we compute the query for all states $s_i \in S$ and multiplicate the probability of being in the state with the result of the query. In this context, $X(s, \varphi)$ will compute the result for $X(\varphi)$, assuming the current state is state s.

Now we can create a vector of probability distributions defining the probabilities results for $X(\varphi)$ for all states

$$\Omega = \{s_1, s_2, s_3, \cdots, s_n\}$$

with $p_i \in \mathbb{R}$ and the results for the operator $X(\varphi)$ for all states s

$$X : \omega^n \to [0,1]^n, \omega \in \Omega$$

Then we can compute the vector $X(S, \varphi)$ for all states with

$$X(S, \varphi) = S \cdot X = \begin{cases} s_1 \cdot X(s_1, \varphi) \\ s_2 \cdot X(s_2, \varphi) \\ \vdots \\ s_n \cdot X(s_n, \varphi) \end{cases}$$

Of course we only need to compute $X(s_i, \varphi)$ for states s_i with a probability of being the active state $S_i > 0$. We can compute the other operators in the very same way. Of course the tradeoff here is adding complexity for the computations of the context predictions while at the same time gaining information about uncertainty.

6.6.9 Optimizations to existing Model Checking algorithms

There is hardly a way of using caches in order to avoid the recomputation of queries since with every context update caused by a context switch the transition system changes and can possibly make all cached query results stale. Also the Next operator has very low demands on computing power so we do not see a need of further optimization here.

One problem though in Model Checking is that especially for queries with the Until operator computations come with a higher overhead than with the Next operators. In order to compute the result of the Until operator we perform a depth first search. When

performing a lot of those queries this can easily get expensive in terms of computing power. Some of those queries could be avoid if it was previously known if a context is reachable at all from a starting location, because in this case the operator would return zero probability.

We propose to solve this problem by adding a connectivity matrix to our context predictor. This connectivity matrix incorporates the transitive closure of the relation $P(s, s')$. Since the transition system is built step by step and transitions between states are also added step by step we can simply update the connectivity matrix with every transition added.

We explain this by an example: When receiving a context update with a new transition from state s to s' to be created we add a reachable flag for every state s'' to state s' where the reachability of s'' and s is already given. This can easily be performed by the Floyd-Warshall algorithm for graph analysis.

This way before every Until query is computed first a check can be made if there is connectivity at all. If there is no connectivity between the related states the computation can be skipped and immediately false can be returned. The computation of the connectivity matrix is computationally very inexpensive and can be performed online during run-time, therefore also complying with our requirements from chapter 4.

Chapter 7

Evaluation

In this chapter we describe what kinds of methods we used in order to evaluate our approach. This includes tests for deriving properties from the transition systems generated by our real world data sets. It also includes metrics and a description for the evaluation of the different operators of our extended PCTL query language.

7.1 General approach

The evaluation process consists of a number of steps being performed for each user of the whole data set:

- read in data file(s) for one user (in case of the Mainkofen data set possibly multiple data files), this leads to a transition system we will use as the simulation model
- evaluate the generated transition system
- take over the transition system used by the simulation model for the predictor model (in case of the Mainkofen data set also generate more time stamps for the DPDFs)
- generate a number of random walks using the simulation model and the corresponding parameters
- evaluate all operators with the generated random walks using the predictor model
- save evaluation data for later use, e.g. creation of diagrams

In order to perform an evaluation for our operators we perform multiple random walks on our transition systems. Each random walk consists of a number of states and a number of transition times that lead from state to state. For the next Operator $X(\varphi)$ and its siblings we perform an evaluation at every single state of every random user trace which has at least one successor state (this also means we always ignore the last state of a user trace). This ensures that there is actually always at least one state following the current state, giving us results for the Next operator.

The evaluation of the Until operator $\Phi U^{T \leq t \leq T+\Delta T} \Psi$ is handled in a slightly different way: since the evaluation part is computationally quite costly because of the way we evaluate the operator we only evaluate the very first state of every random user trace.

7.1.1 Generation of random user traces

In order to evaluate our operators we simulate user behaviour by generating randomized user traces from our simulation model. This is accomplished by performing random walks

over our transition systems obtained through the input of real world data.

The starting state for each random walk is randomly chosen from all states in the simulation model. The next transition taken is also chosen randomly using the roulette-wheel algorithm explained in detail below. The transition taken leads to a new state where we repeat the process of selecting a new transition. In total we perform a number of n random walks over our simulation model and record all states we pass and the times it takes to go from state to state. Which transitions we take is computed by the roulette wheel selection described below. The transition times compute by taking the mean μ of all transition times of the corresponding DPDF and its variance σ and then generating a new random Gaussian from $N(\mu, \sigma^2)$. We stop each random walk either when we reach an absorbing state or when we passed k states in the random walk. This results in n randomly generated user traces with a maximum length of k states and $k-1$ transitions.

As already mentioned above, these randomly generated user traces do not influence our simulation model parameters since we do not update the transition probabilites or the DPDFs of the concerned transitions in our simulation model.

Using these randomly generated user traces we can now start evaluating our operators.

7.1.2 Roulette-wheel selection

Originally roulette-wheel selection or fFitness proportionate selection is a genetic operator used in genetic algorithms for selecting potentially useful solutions for recombination. When randomly selecting a future transition to take we face the problem of how to take the transition probabilities into account. I.e. we do not want every transition to have the same probability to be chosen, but to reflect the transition probability related to the transitions.

The basic idea behind the roulette wheel algorithm is to spin an imaginative wheel and watch where the ball lands. Unlike a real roulette wheel in our imaginative roulette wheel the pockets are not discrete, holding only one number, but can have different sizes according to the probabilities related to those pockets.

Example Given a transition system TS with four states $s_i, i = 0, 1, 2, 3$ and the labels $L(s_0) = A, L(s_1) = B, L(s_2) = C$ and $L(s_3) = D$. The transition probabilities shall be $P(s_0, s_1) = 0.65, P(s_0, s_2) = 0.25$ and $P(s_0, s_3) = 0.1$. Since we only look at transition probabilities we shall ignore any data about transition times in this example. The starting state shall be state s_0.

So now in our imaginative roulette wheel we divide the interval $[0, 1]$ into three pockets, so we get a one-to-one mapping of one pocket for each transition. The probabilities of taking a transition are directly mapped to these interval parts. So the first pocket of the interval becomes $[0, 0.65)$, has a size of 0.65 and is mapped to transition T_{s_0,s_1}. The second pocket of the interval would be $[0.65, 0.9)$, has a size of 0.25 and is mapped to transition T_{s_0,s_2}. The last pocket of the interval would be $[0.9, 1.0)$ with a size of 0.1 and a mapping to transition T_{s_0,s_3}.

Now upon rolling the ball i.e. drawing a uniform random number out of $[0, 1]$ we simply check into which of the three pockets the number belongs and select the transition that is mapped to this part of the interval for the next state we wish to choose.

7.2 Performance metrics

In order to evaluate our operators we shall use several metrics displaying several properties about the qualities of our results. To illustrate this we now introduce an example using a pregnancy test for women.

Imagine a test evaluating the pregnancy of a woman. There are only two possible results for the test since each woman either is or is not pregnant. The test outcome can be positive, predicting that the woman is pregnant, or negative, predicting that the woman is not pregnant. The test results for each woman may or may not match the subject's actual status, depending on the quality of the test.

In that setting we can differentiate between four possible outcomes for our prediction and reality:

- True positive (TP): A pregnant women is correctly diagnosed as being pregnant
- False positive (FP): A women is incorrectly identified as being pregnant
- True negative (TN): A women is correctly identified as not being pregnant
- False negative (FN): A pregnant women is incorrectly identified as not being pregnant

When taking a lot of samples from different pregnancy tests of the same product we can use these four outcomes in order to reason about specific quality properties of the pregnancy test. In short we will abbreviate the number of false positives with FPs, the number of false negatives with FNs, the number of true positives with TPs and the number of true negatives with TNs.

For our evaluation we use four different metrics describing various quality properties:

Accuracy Accuracy directly describes what fraction of the predictions have been correct predictions in comparison to all predictions. A pregnancy test with a high accuracy will therefore indicate that the result of the pregnancy test is very likely to describe reality.

Accuracy does not necessarily propose though that if the pregnancy test is positive the error must be small. The error for a positive test result could be very high but at the same time the error for a negative test result could be very low. Accuracy does not capture these side effects, basically it averages them out.

$$Accuracy = \frac{TPs + TNs}{TPs + TNs + FPs + FNs}$$

Precision Precision directly describes what fraction of the predictions that predicted a positive result have been true in comparison to all predictions that predicted a positive result. So when a test has a high precision we can directly derive that the test hardly ever returns a wrong result if the result is positive. Applied to our pregnancy test example: If a pregnancy test has a high precision this means that when indicating a pregnancy it is very likely for the woman taking the test to actually be pregnant. It does not say anything though about the quality of the result if the pregnancy test says the woman is not pregnant.

$$Precision = \frac{TPs}{TPs + FPs}$$

Recall or sensitivity Recall describes what fraction of our predictions that predicted a positive result has actually been true in comparison to the number of predictions with a positive result that in fact are correct and the number of predictions that were negative but in fact were wrong. This puts the correct positive predictions in comparison to the wrong negative predictions. Getting back to our analogy with the pregnancy test this means that a high recall value means that there are not many predictions that falsely predict that a woman is not pregnant.

$$Recall = \frac{TPs}{TPs + FNs}$$

Specificity Specificity describes what fraction of our predictions that predicted a negative result have actually been true in comparison to the number of predictions that in

a correct negative result and the number of predictions that resulted negatively but were in fact wrong. This puts the correct negative predictions in comparison with the false positive predictions. Again going back to our pregnancy test this means that a high specificity value means that if the pregnancy test returns in a negative result with a very high likelihood the woman taking the test is not pregnant.

$$Specificity = \frac{TNs}{TNs + FPs}$$

Conclusion All metrics specialize in different means on different properties for the prediction results. In order to gain the best results for our predictions we should strive to maximize all four metrics. In general, depending on the transition system we will examine it is not always possible to get a very good result for all metrics. There might be tradeoffs involved between different metrics or between the metrics and system properties, e.g. the granularity of interval sizes Δt versus good results for the metrics.

7.3 Real world data sets

The real world data we are using in our evaluation is provided in text or Microsoft Office Excel files. In both cases the user traces are organized in lines. In every line there is the current time and values for the according atomic propositions. The next line then contains the next context the user experienced. The end of the file would then signal the end of the trace.

A simple example:

Let us assume we choose a time interval Δt of 10 minutes for this example.

Line	Time	Location	Activity
1	0	dining room	have breakfast
2	15	bathroom	shave
3	20	car	drive
4	40	office	work
⋮	⋮	⋮	⋮
14	620	car	drive

So in this example the user would be in the dining room, having breakfast at the starting time. After 15 minutes breakfast is finished and he would go to the bathroom and shave. Five minutes later he would finish shaving and go to work at his office, driving with his car. Notice that hours later he would return from work again driving in his car.

7.3.1 The Mainkofen hospital data set

Mainkofen project - Peripheral Healthcare Documentation

The Mainkofen project investigates to what degree automatic activity recognition can support the use of hand held mobile devices for nursing documentation. The core idea is that by reducing the documentation to confirming a list of automatically recognized activities the documentation process can be done during periods of "forced low intensity activity" such as walking from patient to patient or waiting for the coffee machine. The

project also studies the impact of activity recognition errors. This project was performed at the Mainkofen state mental hospital in Germany.

The Mainkofen data set is made up of the user traces of nurses working in a hospital. The traces include detailed information about the locations and actions of the nurses as well as timestamps when the actions or location changes occur. The nurses's actions are provided in different levels of detail. We shall refer to the different levels of granularity with $Activity - Concrete$, $Activity - SemiAbstract$ and $Activity - Abstract$.

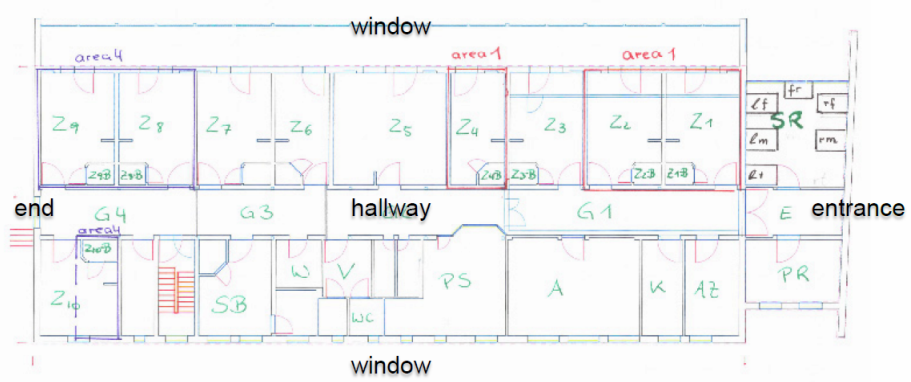

Figure 7.1: The floor plan for the Mainkofen data, as found in [25]

$Activity - Abstract$ supplies us with very general and high level information about the activities of a nurse. Examples for this would be "morning hygiene", "distribute medication", "morning examinations" or the activity "all day", which describes all other activities of the nurse that do not fall into any of the other categories.

$Activity - SemiAbstract$ goes into more detail here. The activity "morning hygiene" in $Activity - Abstract$ for example is described in a number of activities that are far more detailed in $Activity - SemiAbstract$: This includes activities like "measure blood sugar", "take blood sample" and "measure blood pressure".

$Activity - Concrete$ provides us with a very detailed description of what each nurse is doing. For example the $Activity - SemiAbstract$ action "take blood sample" is now even more detailed: "withdraw blood", "put labeled bloodsamples into storage box" and "put blood sample kit back".

Since we are interested in how the granularity of user actions affects the results of our

evaluation we take these three different abstraction levels of user actions into account.

10 rooms, 20-22 patients, 1-3 weeks of stay per patient 4 areas, 4-6 patients per area, 4-5 nurses asigned to each area, one nurse per area present in each shift

25 days of measurment Nov 11 Dec 18 22 workdays 3 saturdays 50 datasets -¿ 2 DS/day, 25 labled, 25 unlabled

6.30 ap. 9.30: Start: after change of shift, End: after breakfast, Around 4 patients per data set (8 a day)

Recorded/labeledtasks: Morning examinations, Morning hygiene, Breakfast

7.3.2 The Dartmouth wireless data set

The Dartmouth wireless data set is a collection of nearly continuous records over three years showing the location (via the associated wireless access-point) of each wireless device seen on the Dartmouth college campus. This dataset has been used in many papers in order to evaluate location predictors [2].

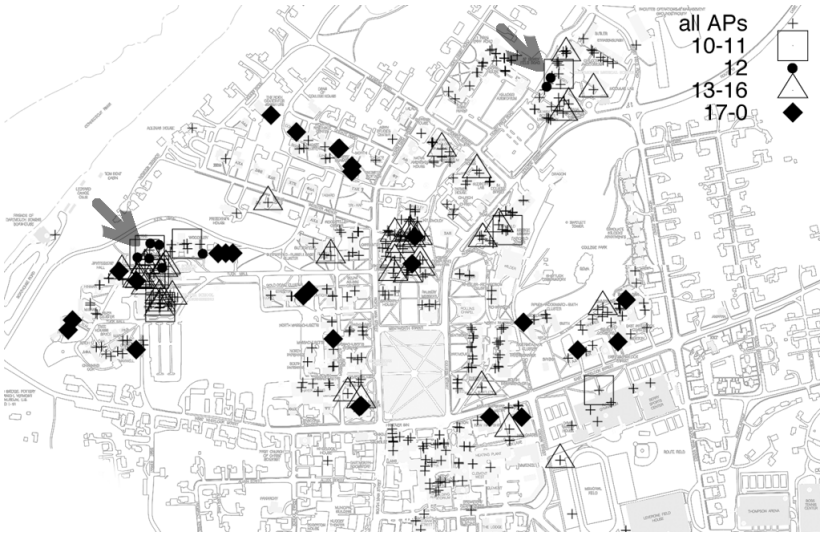

Figure 7.2: Wireless access points on the Dartmouth college campus, as found in [2]

The user traces were extracted from the syslog daemon. Each user's trace is a series of timestamps and locations, i.e. wireless access-point names. The traces vary widely in

length. The length of a trace directly corresponds to the number of locations visited by a user. Users with longer traces were either more active a.k.a. using their card more, more mobile a.k.a. changing access points more often, or used the network for a longer period.

7.4 Evaluation of transition systems

We evaluate different properties for the real world data sets:

7.4.1 State space

One very important property we will evaluate is the number of states being used in the transition systems generated. This gives us insight into how well our Model Checking algorithms will perform since their performance is mostly governed by the size of the state space. This will also enable us to identify combinations of atomic propositions that look promising to evaluate. A higher number of states here translates into more detail for the predictor but can also have the drawback of being too detailed and therefore too specific in order to make any predictions. A lower number might suggest less detail but depending on the reasons for a lower number of states the prediction results might be better.

Compared to a transition system with lots of states the predictions should in general be better with fewer states. In order to evaluate the size of the state space we simply sum up the number of unique states we get in all of our transition system after reading in each user trace.

In figure 7.3 we see the number of different states in the different transition systems generated for each setup of atomic propositions. The atomic propositions we used for the construction of the transition systems are $location$, $patient$, $Activity - Concrete$, $Activity - SemiAbstract$ and $Activity - Abstract$. We also used several combinations of atomic propositions in order to provide more information like $location \times patient$, $location \times Activity - Concrete$, or $location \times Activity - Concrete \times patient$. The states refer to the total amount of different states over all user traces. Since the states refer to the cross product of its atomic propositions we can observe that the state space increases by large

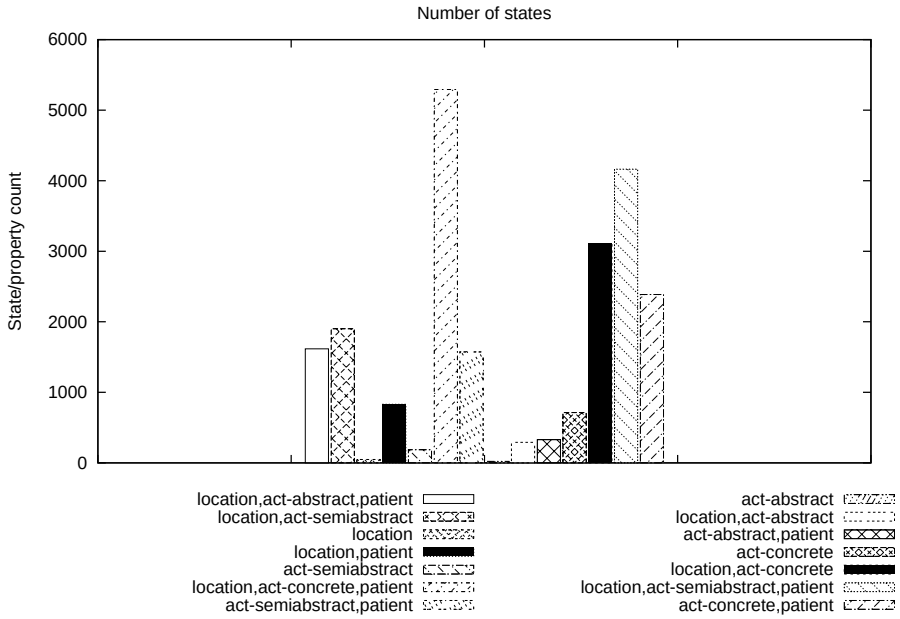

Figure 7.3: State/property count in the Mainkofen data set

amounts when adding another atomic proposition to the evaluation setup.

At the first look the number of states looks surprisingly much for $Activity - Concrete$, but we need to take into consideration that $Activity - Concrete$ displays very fine grained information about what each nurse is doing. In contrast to this $Activity - SemiAbstract$ describes the actions of the nurses in a more coarse-grained way and finally $Activity - Abstract$ describes only the very basic types of activities.

If we want to take into account as much information as possible we should therefore use the atomic propositions $location \times Activity - Concrete \times patient$.

In the Dartmouth data set we only have one atomic proposition, the location of the user, which translates into an access point on the college campus. Therefore we do not provide a diagram but only the number of different states. In total there are about 580 unique states in the Dartmouth data set.

7.4.2 State properties

In this part of the evaluation we take a close look at how many successor states each state has over all transition systems generated by user traces. The more states exist with a high number of successor states the more random the graph will look like. The more states exist with a small number of successor states the more determininistic the transition system will be. Of course the number of successor states itself does not give us any information about the distribution of transition weights. Therefore it is possible for a state to have many successor states and only one of them has a very high transition weight in comparison to all other successor states. That's why we also take a look at the distribution of transition weights.

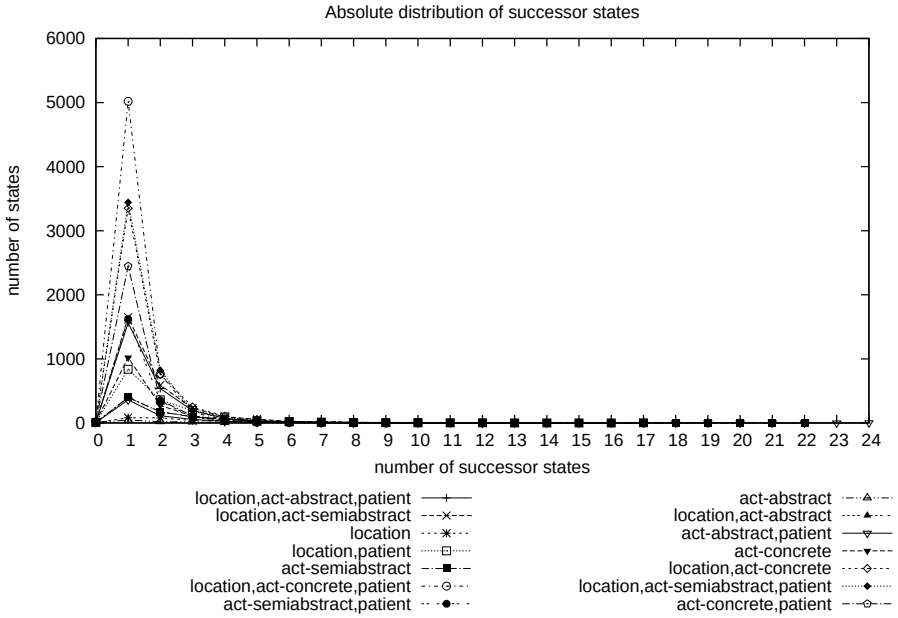

Figure 7.4: Absolute distribution of outgoing edges in the Mainkofen data set

Taking a look at the distribution of successor states displayed in figure 7.4 we notice that the fine granularity provided by using more atomic proposition in conjunction with each other is reflected in the transition system in more states that have only one or maybe two successor states. This creates a lot of very deterministic paths in the transition system. An example for this kind of transition system would be the transition system with the atomic propositions $location \times Activity - Concrete \times patient$. At the same time we observe that the transition system generated by using only the atomic proposition *location* behaves

more randomly - it has many states that have far more than two successor states.

I.e. this means that activities like taking a blood sample that actually consist of a number of smaller steps in between form chains of states. Therefore using a more concrete detail level for the atomic proposition $Activity - Concrete$ we get exactly the opposite of a random graph, closely connected areas that mostly behave like a long chain of actions. This makes a lot of sense - e.g. when taking a blood sample a nurse is very unlikely to interrupt the process and start doing something completely else.

In contrast to this we have a transition system with only the atomic proposition of *location* that behaves in a more random and nondeterministic way. This can also be explained: A nurse usually has some kind of schedule for her tasks and a list of patients she has to take care of. But unlike the task list she has freedom to take care of her patients in an order that benefits all patients and that can be adjusted daily. I.e. one day she might go to room A first and then to visit room B and later room C. But because the patient in room C needs special care the next day she might reverse her schedule and go from room C to room B and then to room A, performing the same activities in the same order, but just in a different room.

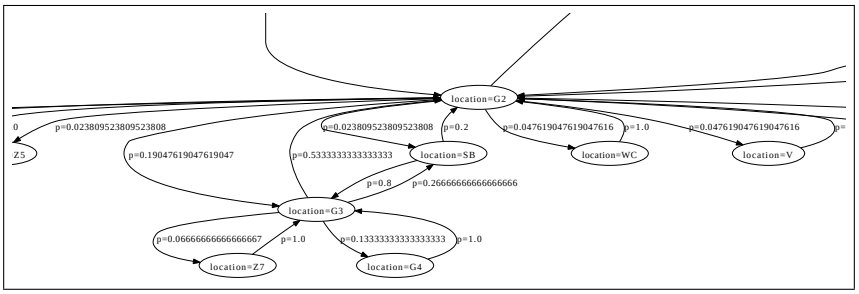

Figure 7.5: Example of a part of a transition system for Mainkofen data using only the atomic proposition *location*

In figure 7.5 we display a part of a transition system generated by using only the atomic proposition *location*. Even though the transition system for the atomic proposition *location* is more random we still see a lot of structure. For example, given the context $location = G2$ there are many other locations to choose from to go to. But when after going from location $G2$ to location WC there is only one possible way to go which is going back to location $G2$. We observe this pattern a lot in the data.

Figure 7.6 also displays this behaviour, but in a cumulative setting. The comparison of the different combinations of atomic proposition with the cumulative exponential distribution shows exponential growth for all curves. In other words the likelihood of having more successor states drops with an increasing number of successor states. I.e. there are a lot of states that have very few successor states. We can even derive from figure 7.6 that more than 80% of a transition systems' states will have less than three successor states in general using the Mainkofen data set.

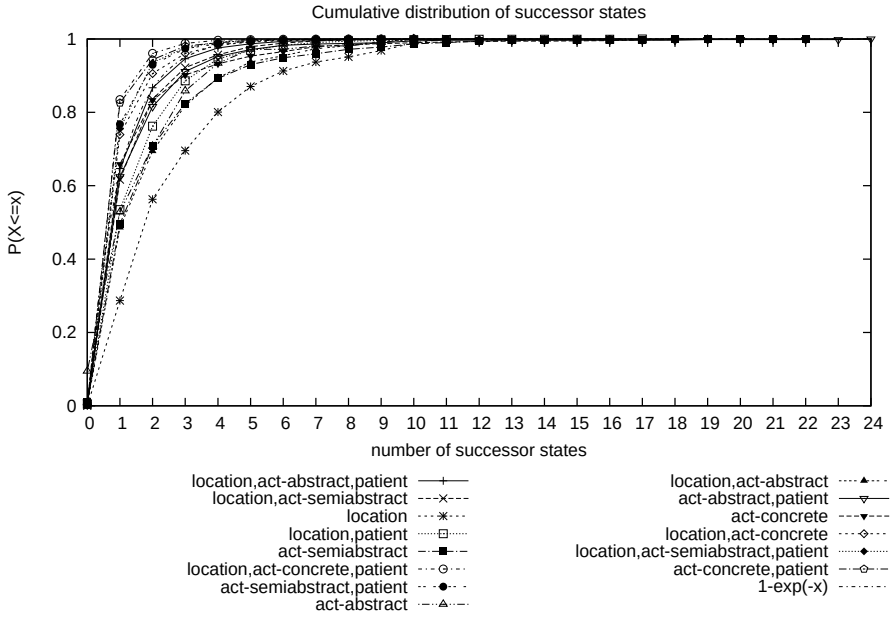

Figure 7.6: Cumulative distribution of outgoing edges in the Mainkofen data set

We can also derive from the diagram that the combination of atomic propositions $location \times activity-concrete \times patient$ and $activity-concrete \times patient$ have the fewest number of successor states over all states. So in these transition systems about 85% of all states have only one successor state, more than 95% of all states have a maximum of two successor states and more than 99% of all states have a maximum of three successor states. We also observe that the atomic proposition *location* clearly has the highest rate of successor states per state: Here only 30% of all states have only one successor state, 55% of all states have a maximum of two successor states and 75% of all states have less than three successor states.

When only looking at the number of successor states it doesn't seem to make any statis-

tically significant difference if we choose the combination $location \times activity - concrete \times patient$ or $activity - concrete \times patient$. This suggests that adding the atomic proposition *location* does not provide any additional information into the transition system. We will look into that later.

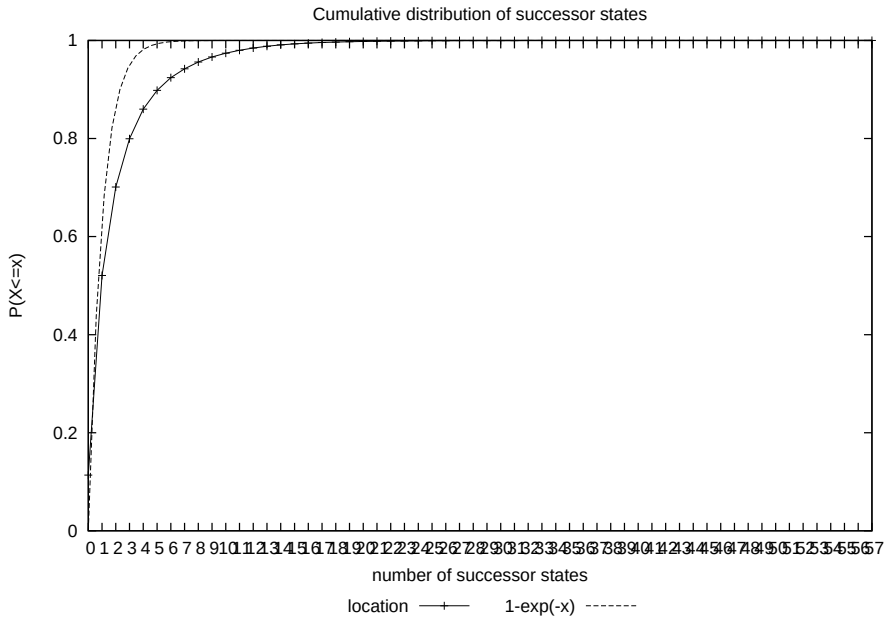

Figure 7.7: Cumulative distribution of outgoing edges in the Darthmouth data set

When looking at the same evaluation for the Dartmouth data set we get similar results. When looking at figure 7.7 we can again see that most states only have few successor states in the transition systems. The big difference here is the sheer numbers - in the Dartmouth data set we can again observe than most states only have one successor. Compared to the Mainkofen data the distribution here is somewhere between the combinations of atomic propositions of $location \times activity - concrete \times patient$ and *location*. In the Dartmouth data we have about 50% of all states with only one successor state, 70% of all states with a maximum of two successor states and 80% of all states have less than three successor states.

This could mean that with the Meinkhofen data set we could in general get better results in our evaluation part for our operators. Of course this assumption does not take the distribution of transition probabilities into account. We provide an in depth study of the transition probabilities in the next section.

7.4.3 Transition properties

First we will take a look at how the transition probabilities are distributed in the different data sets. Going back to the previous section we discussed that we expect transition systems in which states in general have more successor states to be behave less deterministic than transition systems in which states have less successor states. We will now check if this assumption actually holds true.

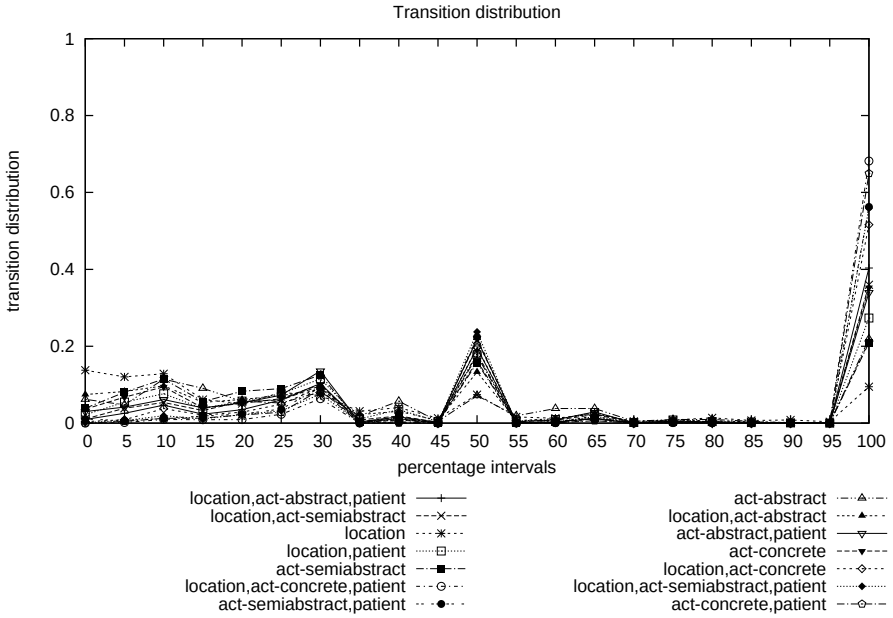

Figure 7.8: Relative distribution of transition probabilities in the Mainkofen data set

In figure 7.8 we see the relative transition probability distribution over all transition systems. The diagram is interpreted in the following way. The tics on the x-axis always stand for an interval of transition probability. So the label 0 on the x-axis actually stands for the interval $[0, 5)$, the label 50 stands for the interval $[50, 55)$ and in general label k stands for the probability interval $[k, k + 5)$. On the y-axis we note the relative proportion of transitions that have their corresponding transition probabilities in the specified intervals.

Taking a look at the atomic proposition *location* we observe that this transition system has a lot of transitions in the intervals $[0, 10)$, even more than in the intervals $[50, 55)$ or $[100, 105)$. No probability interval has more than about 15% of all transitions. This

confirms our assumption; the transition indeed is more nondeterministic.

At the same time checking on the combinations of atomic propositions $location \times activity - concrete \times patient$ we clearly see that more than 70% of all transitions have a transition probability of 1.0, meaning only one successor state, 20% of all transitions have a transition probability of 0.5 i.e. two successor states and about 10% of all transitions have a transition probability of 0.3, i.e. three successor states. Again we see our assumption confirmed. An interesting thing again is the added level of detail provided by the atomic proposition *location*. As we can see in the diagram the combination of atomic propositions $activity - concrete \times patient$ has almost the same properties as the combination of atomic propositions $location \times activity - concrete \times patient$. When evaluating our operators in the next section we will check if our predictions benefit from the added level of detail provided by the atomic proposition *location*.

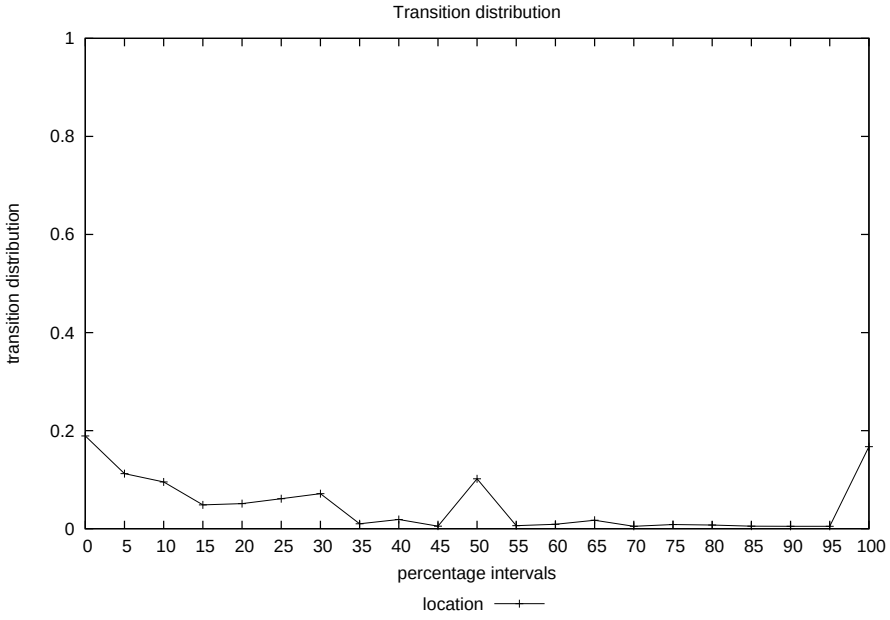

Figure 7.9: Relative distribution of transition probabilities in the Dartmouth data set

When looking at the Dartmouth data set in figure 7.9 we observe a very similar behaviour as in the Mainkofen data set with only the atomic proposition of *location*. No percentage interval contains more than 20% of all transitions and the interval $[0, 10)$ actually contains more transitions than the intervals $[50, 55)$ or $[50, 55)$. So the transition systems generated by the Dartmouth data are less deterministic and therefore will probably lead

to predictions of lesser quality.

Of course there is a third factor that comes into play for the predictions which is the time already spent in the current state. This can in theory make a big difference for the quality of our predictions. We will take a look into this in the next section.

Now we take a look at when context switches actually take place in time. This is important in order to determine time boundaries, time window sizes and dwell times in our operator evaluation later.

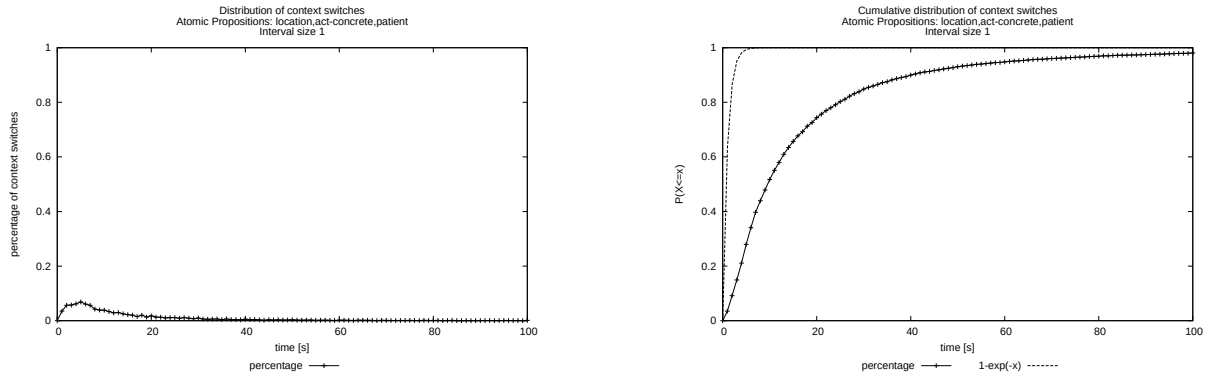

Figure 7.10: Relative and cumulative distribution of transition times in the Mainkofen data set

Figure 7.10 displays all state changes from state to state relative to the total number of state changes.

The diagram on the left displays the relative distribution of context switches in different time intervals. For this diagram we took all transitions from an example transition system generated by the combination of atomic propositions $location \times activity - concrete \times patient$, though we could have chosen any other combination since the transition times are not affected by using a different set of atomic propositions. For the interval resolution of our transitions' DPFDs we chose a value of 1.

The right diagram in figure 7.10 displays the cumulative distribution of context switches in time intervals. Again we chose the combination of atomic propositions $location \times activity - concrete \times patient$.

Examining the left diagram we can observe that most state changes occur during the first

20 seconds, with a smaller tail of state changes between 20 and 40 seconds. Since in this figure we again are looking at the finest grained level of activity this makes a lot of sense, because the level of detail forces the state changes to happen rather often. Also there are hardly any activities using this level of detail that take a very long time to complete. Therefore we can now assume that the time interval $[0, 40]$ will be interesting area for time windows and dwell times for most transitions.

When taking a look at the right diagram in figure 7.10 we can derive that more than 75% of all context switches from state to state occur between 0 and 20 seconds and that more than 85% of all context switches occur within the first 40 seconds. This indicates that limiting the time windows and dwell times for our operators to a maximum of 40 seconds makes a lot of sense for the evaluation part. This figure also shows that the curve behaves more like a square root function than an exponential function.

Although this figure is sampling all transitions in the whole transition system this already points to maybe upcoming problems using continuous time Markov chains, since they are limited to negative exponential distributions for the context switch times: One property of this distribution is memorylessness - time spent in the actual state is not being taken into account when selecting a successor state for predictions. We will take a closer look at transition times for some sample transitions.

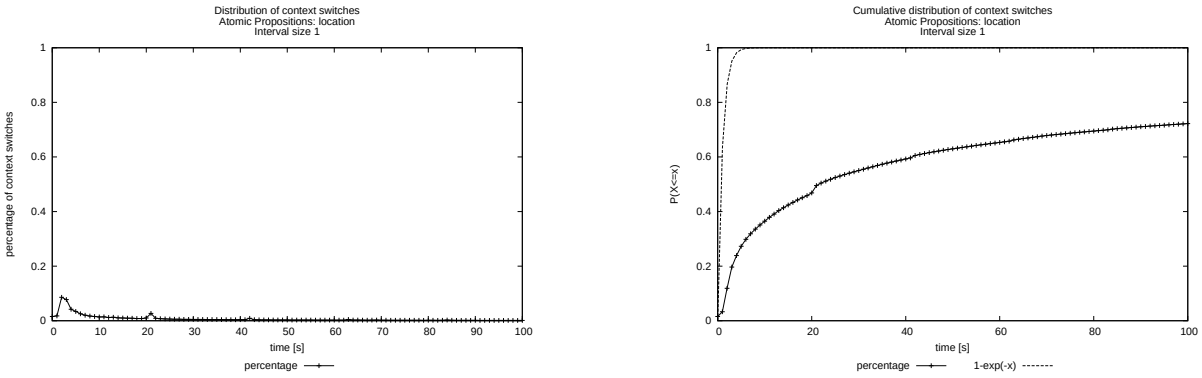

Figure 7.11: Relative and cumulative distribution of transition times in the Dartmouth data set

Figure 7.11 displays the same information for the Dartmouth data set. The resolution for the intervals of the transitions' DPDFs here is 1 second for the left diagram and 60 seconds for the right diagram.

Examining the left diagram we can observe that in the very beginning many of context switches occur within about the first 10 seconds with a peak around 2 seconds. So about 30% of all context switches actually happen during the first 5 seconds. Examining the diagram on the right we observe that the rest of the context switches happens quite uniformly distributed with an emphasis on shorter timespans, i.e. the less time has passed the more likely a context switch is about to occur. Therefore about 90% of all context switches occur within the first 2 hours after entering a new location.

Now in order to support our argument of not using continuous time Markov chains we take a closer look at some example transitions. Unfortunately the Mainkofen data set is not of a sufficient size to draw any conclusions about the nature of time stamp distribution for context switches. Most transitions' DPDFs do not contain more than 5 timestamps. We can still take a look at them and discuss their properties and use them to derive some new information.

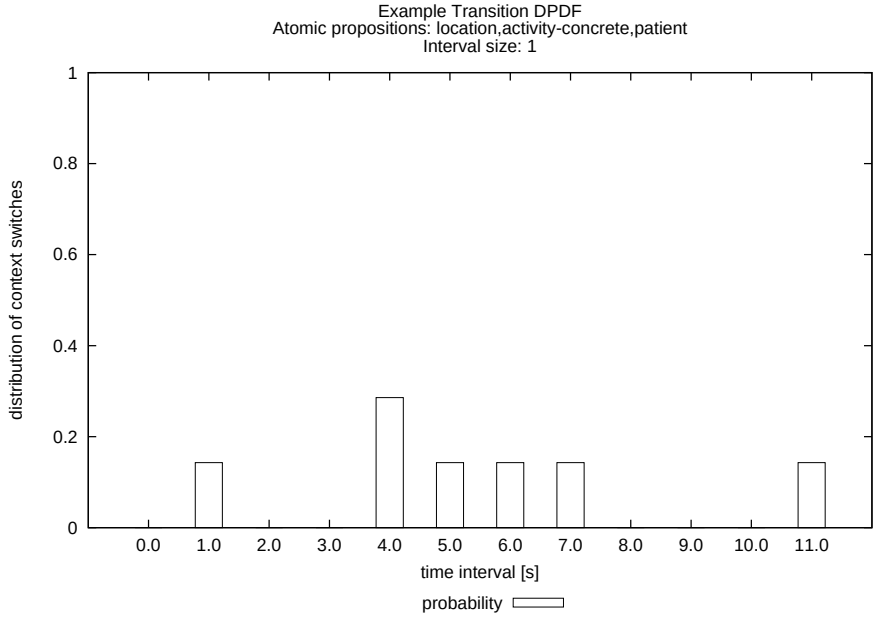

Figure 7.12: DPDF of an example transition from the Mainkofen data set

In figure 7.12 we take a look at an example transition's DPDF from the Mainkofen data set. As we can clearly see it does not have the looks of a negative exponential distribution. The distribution looks more like a Gaussian distribution with a mean $\mu \approx 4$.

Of course the low amount of data means we can not make any statistically significant

assumptions. When taking a look at the top ten transitions with the most time stamps in their DPDFs though we can observe that none of the distributions looks like a negativ exponential distribution. Despite the low amount of timestamps we are being provided with by the data set many transition share this property. It also seems reasonable to assume a normal distribution for the duration of human tasks. For this reason we chose to use this information to create new timestamps for the transitions in the Mainkofen transition system in the following way: We generate new data for the transitions by generating timestamps randomly drawn from $N(\mu, \sigma^2)$ with $\mu = \frac{1}{n} \cdot \sum_{i=1}^{n} t_i$. In order to keep the probability ratio $P(s, s')$ constant between different transitions we generate $c = k \cdot w_{s,s'}$ timestamps for each transition between all states s, s' with a constant factor k that is provided as a simulation parameter. Using a constant factor for the generation of simulation data will ensure that the probability ratios between all transitions from state s to his successor states s' will remain constant.

7.4.4 Conclusion

Depending on what level of detail we choose for the atomic proposition *activity* and which other combinations of atomic propositions we choose we get a mostly deterministic or a very random transition system. Of course this will affect the quality of our predictions. The quality is affected in two ways: We expect the results in a less detailed activity model to be more accurate, since the model does not make a difference between different actions of the same kind as long as they are of the same kind. In a more detailed activity model this might create prediction errors since it might not be 100% clear whether to take the blood sample first or to measure the blood pressure before. So in this context we expect lower values in our evaluation metrics displaying a worse prediction performance. At the same time though we know that one property of transition systems using a high level of detail for activity is that they are highly deterministic, leading to only few states that have more than 2 or 3 successor states. This should influence our evaluation results in a rather positive way.

We may also assume that the distribution of time stamps on a transition's DPFD is not necessarily a negative exponential distribution but may follow any distribution. This is a strong argument for using Semi-Markov chains instead of continuous time Markov chains.

This allows us to capture the data quite closely. Since the number of user traces was not big enough to perform an adequate evaluation in terms of transition time stamps them we chose to use these traces as a simulation model from which we automatically generated a satisfying amount of new traces.

7.5 Evaluation of operators

This section deals with finding system parameters that promise to return good results for the use of our operators. This mainly consists of finding a suitable probability threshold p that delivers good results for all four metrics. Of course also time windows and dwell times are an important factor, we want to remind the reader that we found suitable time parameters in the previous section.

In order to perform the evaluation we are given a starting state s and a follow up state $s' \models \varphi'$, both from a randomly generated user trace. In the transition system we also have a set of successor states of s which we call $S = \{s'' | s'' \in succ(s), s'' \models \varphi''\}$. The time spent in s before switching contexts to state s' shall be t_{real}.

7.5.1 $X(\varphi)$

In order to evaluate this operator we first need to remind the reader of the fact that this operator does not take time into account at all. It only takes into account the transition probabilities based on the transition weights. For more details on this operator please refer to section 6.4.

Now for each state $s'' \models \varphi''$ that the starting state s leads to we calculate $p = X(L(s''))$. Now for all $q = \frac{k}{10}, k = 0, 1, 2, \ldots, 10$ we check if $p \geq q$. If that is the case we predict the corresponding state to be the next state. If $p < q$ holds we predict the state is not going to be the next state. Now we compare our prediction with the actual next state. This will lead to a FP, TP, FN or TN in the following way. If the predictor predicts a state to be the next state and the next state is actually this state we count the result as a TP. If the predicted next state is not the next state though we count the result as a FP. In case we predict a state not to be the next state and it is in reality not the next state the

result will be a TN. The last result is a FN which occurs when we predict a state as not being the next state when in reality it is the next state. This method is used throughout the whole evaluation part for all operators unless otherwise mentioned.

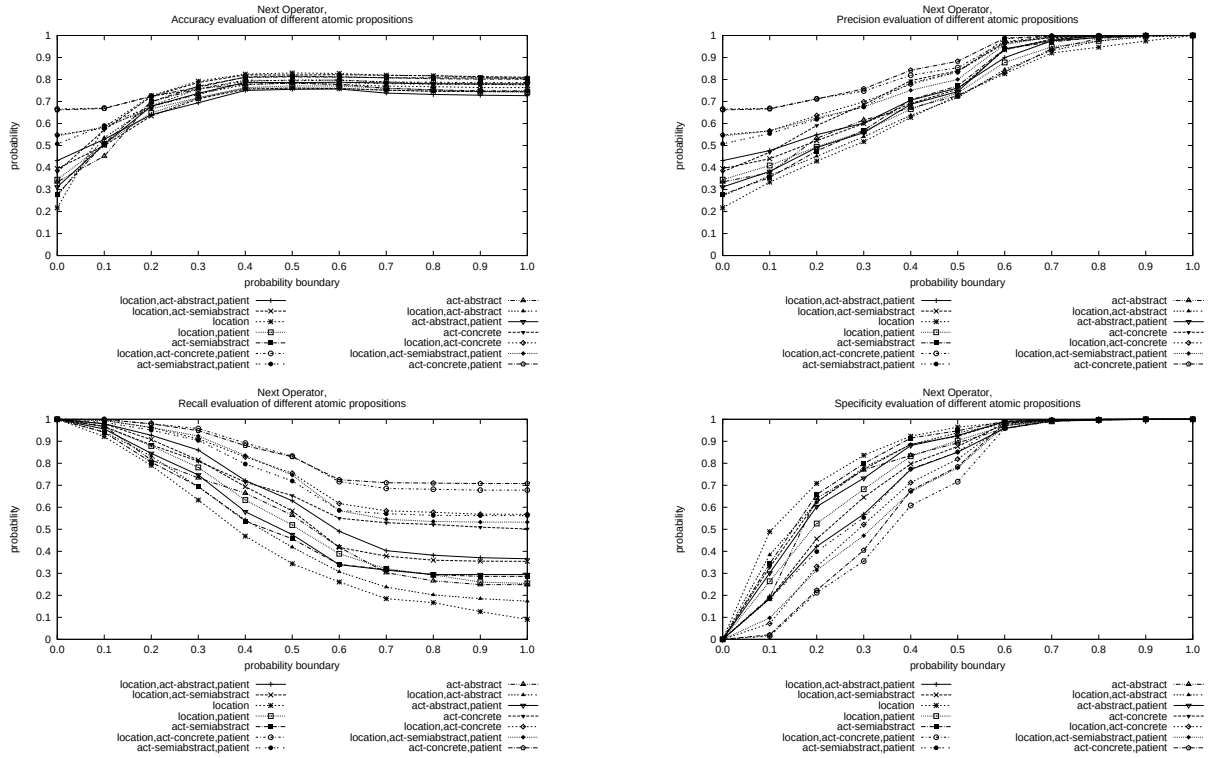

Figure 7.13: Metrics for $X\,(\varphi)$ in the Mainkofen data set

Accuracy evaluation The results displayed in figure 7.13 show the different accuracy values for different probability thresholds. In general accuracy describes the fraction of correct positive and correct negative predictions in comparison to all predictions. We remind the reader to look up the exact definition of accuracy in section 7.2.

We get the best accuracy results for a setting of $0.4 \leq p \leq 0.6$, for $p \geq 0.6$ the results are slightly worse. For $p \leq 0.4$ the results are noticeably worse for all combinations of atomic propositions.

When choosing $p \geq 0.4$ we have an accuracy of betwen 70% and 80%, that is 70 to 80 out of 100 times the predicted result when predicting that a state will be the next state or when predicting the opposite, that a state is not going to be the next state. That is

already a very good result.

The variation between different combinations of atomic propositions is not very high. Overall there is a 10% difference in accuracy when comparing the combination of atomic propositions with the highest accuracy values and the combination with the lowest values. This suggests that for this operator the different transition systems do not have a huge impact on the outcome.

Taking a look at the combination of atomic propositions $location \times activity - concrete \times patient$ we note that compared to all other combinations of atomic propositions the accuracy results are the worst results in the whole diagram with a value of roughly 0.72. Also conversely the results for the atomic proposition *location* are the best with a value of about 0.8. These two facts only hold true for a threshold of $p \geq 0.4$ though. For $p \leq 0.4$ we get very different results; The combination of atomic propositions $location \times activity - concrete \times patient$ actually performs quite well in this setting starting with a value of 0.65. Here we get the opposite result for the atomic proposition *location* with a value of only 0.21.

This seems to reflect facts based on the different properties of the transition systems. I.e. with the combination of atomic propositions $location \times activity - concrete \times patient$ there are hardly any transitions with a transition probability of less than 0.3. In the transition system generated with the atomic proposition *location* this is not the case, that is why for lower values for p the results are not as good.

Precision evaluation The precision evaluation in Figure 7.13 shows the same kind of information as in the accuracy evaluation, but with a different evaluation metric. This diagram uses precision as evaluation metric. As already explained in section 7.2 precision describes the amount of TPs in comparison to the sum of TPs and FPs. This captures the quality of all predictions that predict a state to be the next state.

The bandwidth of different results here is far higher than in the accuracy diagram shown before. Especially for low values of the probability threshold p the differences between combinations of atomic propositions are up to 0.45. Starting from a probability threshold $p \geq 0.6$ all combinations of atomic propositions have a precision value of over 0.8.

We get the lowest values for precision with the atomic proposition of *location*, ranging between 0.21 and 1.0. This means that taking only a nurse's location into account for predictions of future location and no other context information (and no time information either) is not enough to get a high precision, resulting in many FPs and fewer TPs. Since precision does not include any information about the count of TNs or FNs the relatively high values for precision using higher values for the precision boundary p have to be taken with caution.

The highest values for precision are the results for the combinations of atomic propositions $location \times activity-concrete \times patient$ and $activity-concrete \times patient$, the results for the latter combination is even a little bit better. All other combinations of atomic propositions are in between those extremes. This confirms our previously made assumption that in fact adding more information into the transition systems can lead to a predictor of lesser quality. We will later take a look into other operators that also use time for the prediction result in order to confirm or deny this assumption.

Recall evaluation Figure 7.13 displays the evaluation results for the metric recall. Recall basically describes the fraction of TPs in comparison to the sum of TPs and FNs, this describes the quality of true positive predictions when taking the number of wrong negative predictions into account. More details about the definition of recall can be found in section 7.2.

The bandwidth of different results is again very high. Especially for high values of the probability threshold p the results differ by up to 0.6. With $p \geq 0.6$ all combinations of atomic propositions have a recall value of less than 0.7.

The highest values of recall can again be reached by using the combinations of atomic propositions $location \times activity-concrete \times patient$ and $activity-concrete \times patient$. For $p \geq 0.6$ we reach a recall value of 0.7 which is still very good. Interestingly again $activity-concrete \times patient$ performs slightly better than $location \times activity - concrete \times patient$ despite the lack of information given by the atomic proposition *location*. The results again show that there are far more correct positive predictions than incorrect negative predictions. We will examine if time is a factor to change this fact when evaluating time based next operators in the next sections.

The lowest values for recall again are reached by the atomic proposition of *location* with a value of less than 0.2 with $p \geq 0.7$ and going down to 0.1 for $p = 1.0$. This means that with higher values of the probability threshold p the number of incorrect negative predictions is becoming larger and larger. This agains speaks for our assumption that only taking *location* as atomic proposition may lead to bad evaluation results. Of course this operator does not take time into account, so when evaluating time based next operators in the next sections we will check into this issue again.

Specificity evaluation Figure 7.13 displays the evaluation results for the metric specificity. Specificity describes the fraction of correct negative predictions in comparison with the incorrect positive predictions. For a more specific review of this metric please take a look at section 7.2.

Again the spread is very high, at probability threshold $p = 0.3$ the difference between the best performing combination of atomic propositions and the worst combination is more than 0.5. For probability thresholds of $p \geq 0.6$ the value of specificity for all combinations of atomic propositions is higher than 0.95. For all combinations of atomic propositions a higher value for the probability threshold leads to less FPs and more TNs in the predictions.

The clearly worst performing combination of atomic propositions is $location \times activity - concrete \times patient$. This time $activity - concrete \times patient$ performs significantly better. The best performing atomic proposition again is *location*.

Figure 7.14 displays the evaluation results for all different metrics for the Dartmouth data set. Since there is only one atomic proposition we can conveniently display all metrics in a single diagram. In this data set accuracy rises very fast to over 0.75 starting from a very low probability threshold of $p = 0.1$. For $p \geq 0.1$ the accuracy values stay in the interval $[0.8, 0.9]$. Precision rises much slower with values of greater than 0.8 starting from $p \geq 0.6$. At the same time recall drops linearly with increasing values for the probability threshold p. Specificity rises very fast starting from $p = 0.1$, rising up to more than 0.9 for values of $p \geq 0.3$.

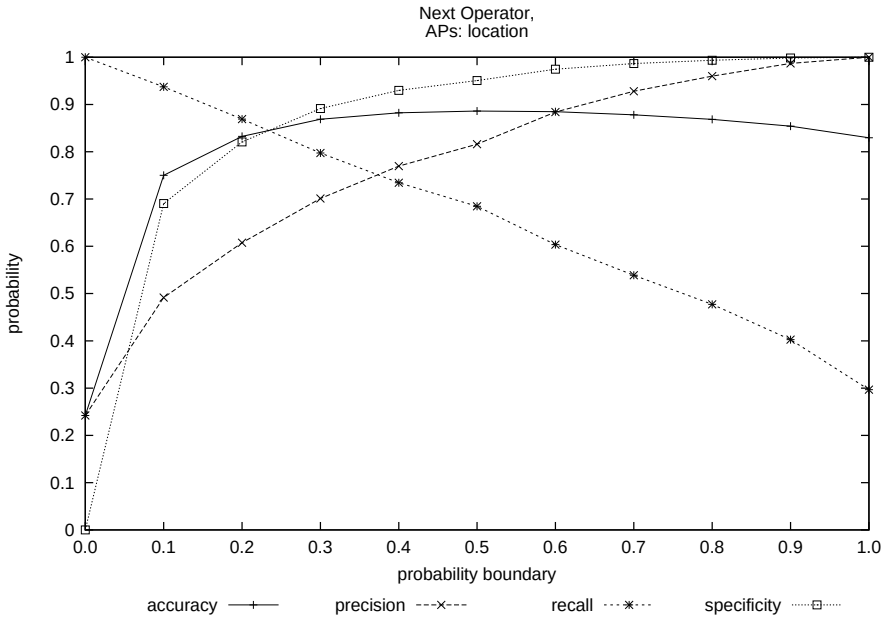

Figure 7.14: Evaluation of $X(\varphi)$ using Dartmouth data set

Conclusion Because of the different properties of the transition systems generated by the Mainkofen data set and the Dartmouth data set the results for this evaluation vary. The biggest difference is that using the Dartmouth data set prediction results are already slightly better for lower values of the probability threshold p.

The best performing combination of atomic properties in total is $location \times activity - concrete \times patient$. In order to maximize the prediction results for the Mainkofen data set for all metrics we should therefore choose a value of $p = 0.5$ for the operator $X(\varphi)$. This ensures a high quality of predictions for all metrics with all different combinations of atomic propositions.

In order to maximize the prediction results for the operator $X(\varphi)$ in the Dartmouth data set we can also choose the probability threshold of $p = 0.5$.

We will use these values for the probability threshold for future comparisons of different operators when using $X(\varphi)$.

7.5.2 $\hat{X}$

We want to remind the reader that this operator does not take time into account. It only takes into account the transition probabilities based on the transition weights. For more details on this operator please refer to section 6.4.

Again we compute $p = X\left(L\left(s''\right)\right)$ for all successor states of s. The operator returns with the probabilities related to taking the specific transitions. We now choose the state that is related to the highest probability we just computed with the regular $X\left(\varphi\right)$ operator. Now for all $q = \frac{k}{10}, k = 0, 1, 2, \ldots, 10$ we check if $p \geq q$, but this time only for the state with the highest probability. If $p \geq q$ we predict the corresponding state to be the next state. If $p < q$ holds we predict the state is not going to be the next state. Now we compare our prediction with the actual next state. This will again lead to a FP, TP, FN or TN.

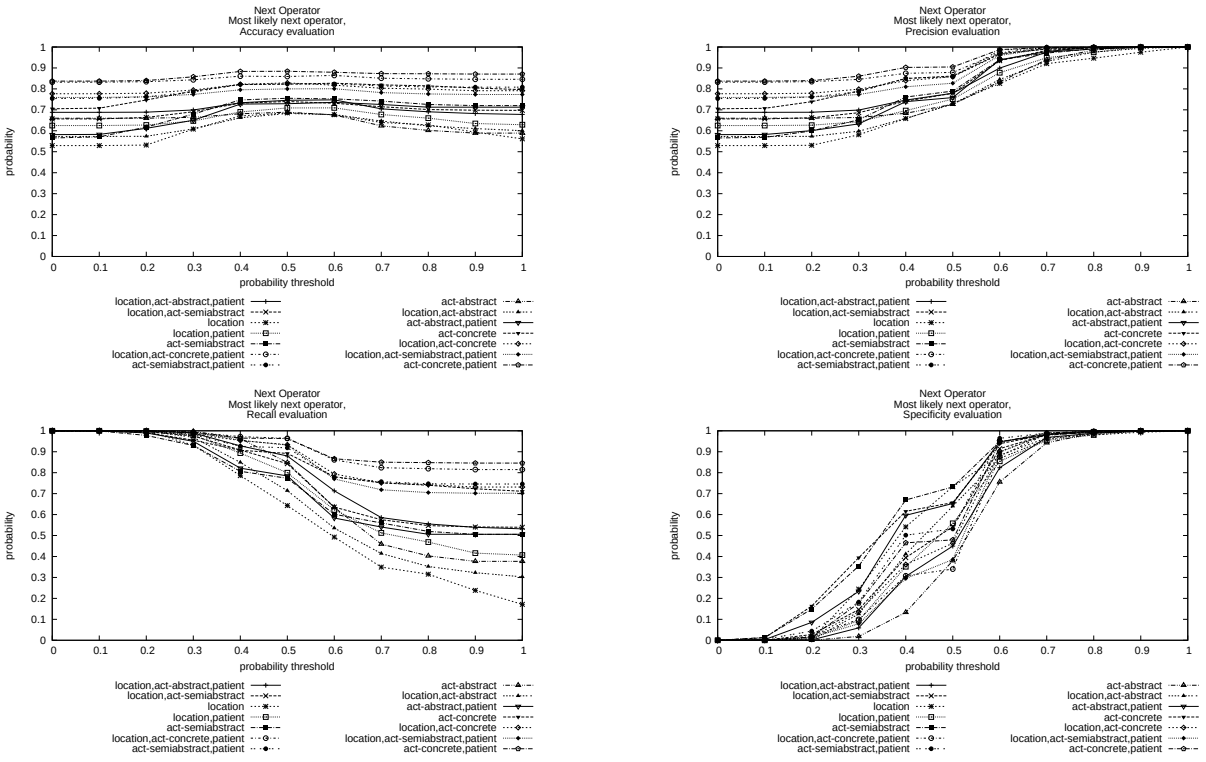

Figure 7.15: Metrics for $\hat{X}$ in the Mainkofen data set

Accuracy Looking at the accuracy diagram in figure 7.15 we see that the results for the different combinations of atomic propositions are very close together. All results are in the interval $[0.5, 0.9]$. We get the best overall results for values of the probability threshold p roughly for $0.4 \leq p \leq 0.7$.

Again the combinations of atomic propositions $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best with values between 0.85 and 0.9. The latter combination also again performs slighly better than the first one. The worst performance is again caused by the atomic proposition *location* with values roughly between 0.5 and 0.7. All other combinations of atomic propositions are in between.

Precision We can witness this behaviour again when examining the precision diagram. Starting from $p \geq 0.5$ the results for all combinations of atomic propositions are higher than 0.7.

The combination of atomic propositions $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best again with values between 0.8 and 1.0. The atomic proposition *location* performs worse with values between 0.5 and 1.0. All other combinations of atomic propositions are in between.

Recall When looking at our evaluation metric recall we can observe that the combinations of atomic proposition $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best with values between 1.0 and 0.8. The atomic proposition *location* performs worst of all combinations again with values between 1.0 and 0.15. Overall we get the best results for values p of the probability threshold $p \leq 0.5$.

Specificity Examining specificity we observe a huge increase of specificity when setting the probability threshold higher than 0.5. For $p \leq 0.5$ we have a maximum of 0.7 for all combinations of atomic propositions, but for $p \geq 0.6$ no combination of atomic propositions is less than 0.7. For $p \geq 0.6$ we get the best results overall with values higher than 0.7.

This time though the best performing combinations of atomic propositions are $activitiy - semiabstract$ and *location* with values over 0.9 for $p \geq 0.6$. The worst performing atomic

proposition is $activity - abstract$ with values of less than 0.9 for $p \leq 0.7$.

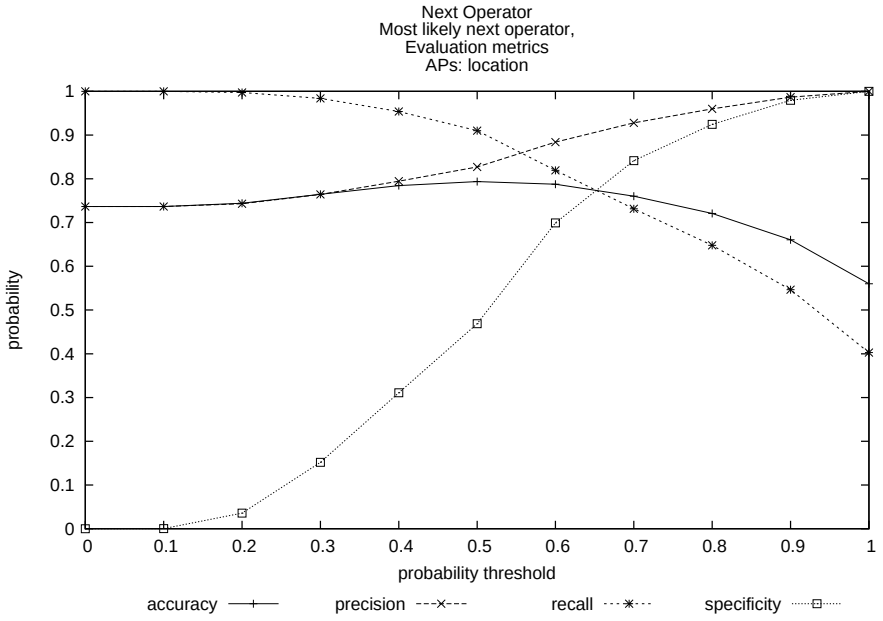

Figure 7.16: Evaluation of $\hat{X}$ using Dartmouth data set

Figure 7.16 displays the evaluation results for all different metrics for the Dartmouth data set. In this data set accuracy stays around a value of 0.7 until the probability threshold exceeds $p \geq 0.9$. Precision already starts out with a value of 0.7, with values over 0.8 for $p \geq 0.5$ and slowly rising to 1.0. At the same time recall drops with increasing values for the probability threshold p reaching 0.4 for $p = 1.0$. Specificity is higher than 0.8 from $p \geq 0.7$. The best setting for the probability threshold p for all metrics here would be $p = 0.6$.

Conclusion In order to maximize the prediction results for the Mainkofen data set for all metrics we should therefore choose a value of $p = 0.6$ for the operator $\hat{X}$. This ensures a high quality of predictions for all metrics with all different combinations of atomic propositions. The best performing combination of atomic propositions in total is $location \times activity - concrete \times patient$.

In order to maximize the prediction results for the operator $\hat{X}$ in the Dartmouth data set we should choose the probability threshold of $p = 0.5$.

We will use these values for the probability threshold for future comparisons of different operators when using $\hat{X}$.

7.5.3 $X^{t \geq T}(\varphi)$

This operator also takes the transition times on the DPDFs into account. This is achieved by offering a dwell time t that has already passed in the current state, from this point in time on we look into the future. For more details on this operator please refer to section 6.4.

The parameter t is passed in as a list of time stamps identifying the dwell times we want to evaluate for this operator. Again, if we switched contexts before the dwell time time stamp, i.e. $t_{real} < t$, we do not compute a result for this prediction but ignore it. We would also never compute this result because we would have already skipped to the follow up state s'.

For each state $s'' \models \varphi''$ that the starting state s leads to we calculate $p = X^{t \geq T}(L(s''))$. Now for all $q = \frac{k}{10}, k = 0, 1, 2, \ldots, 10$ we check if $p \geq q$. If that is the case we predict the corresponding state to be the next state. If $p < q$ holds we predict the state is not going to be the next state. Now we compare our prediction with the actual next state. This will lead to a FP, TP, FN or TN. The parameter t is taken out of a list with different time windows. For each time window we save the number of FPs, TPs, FNs and TNs separately.

Accuracy Looking at the accuracy diagram in figure 7.17 we see that the results for the different combinations of atomic propositions are very close together. All results are in the interval $[0.7, 0.8]$. We get the best overall results for values of the probability threshold p roughly for $p \geq 0.4$.

Again the combinations of atomic propositions $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best with values around 0.8. The latter combination also again performs slighly better than the first one especially for small values of the probability threshold. The worst performance is again caused by the combination of atomic propositions $location \times activity - abstract \times$ with values roughly betweenabove

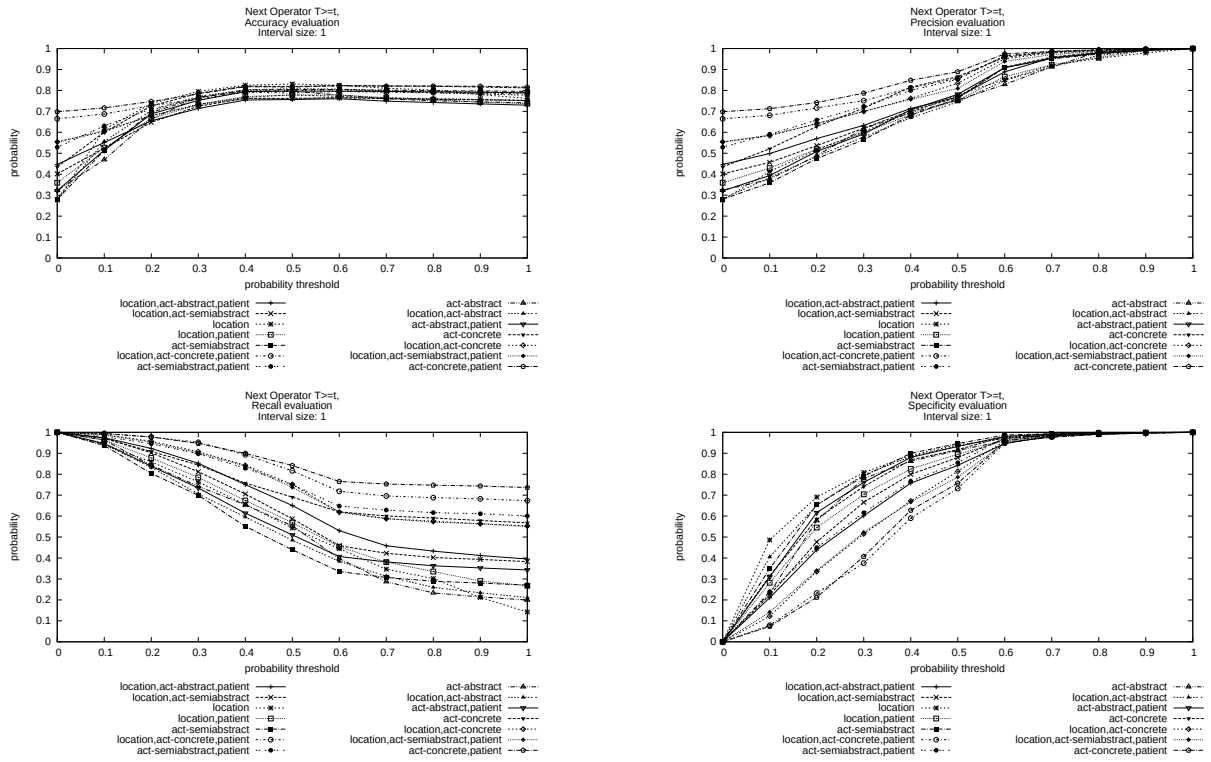

Figure 7.17: Metrics for $X^{t \geq T}(\varphi)$ in the Mainkofen data set

0.7. All other combinations of atomic propositions are in between.

Precision We can witness a very similar behaviour again when examining the precision diagram. Starting from $p \geq 0.5$ the results for all combinations of atomic propositions are higher than 0.7.

The combination of atomic propositions $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best again with values between 0.8 and 1.0. Also again $activity - concrete \times patient$ performs slightly better than $location \times activity - concrete \times patient$. The atomic proposition $activity - abstract$ performs worst. The best results are for the parameter probability threshold starting from $p \geq 0.5$.

Recall When looking at our evaluation metric recall we can observe that spread between different combinations of atomic propositions is wide. The difference is as much as 0.65 for $p = 1.0$.

The combinations of atomic proposition $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best with values between 1.0 and 0.7. Again $activity - concrete \times patient$ performs better than $location \times activity - concrete \times patient$. The atomic proposition $activity - abstract$ performs worst of all combinations again with values down to 0.15. Overall we get the best results for values p of the probability threshold $p \leq 0.5$.

Specificity For $p \leq 0.5$ we observe some spread between the different combinations of atomic propositions, but for $p \geq 0.6$ all results are grouped tightly together. This time the results are reversed - the atomic proposition *location* scores the best results and $location \times activity - concrete \times patient$ scores worst. The best results are attained for $p \geq 0.5$.

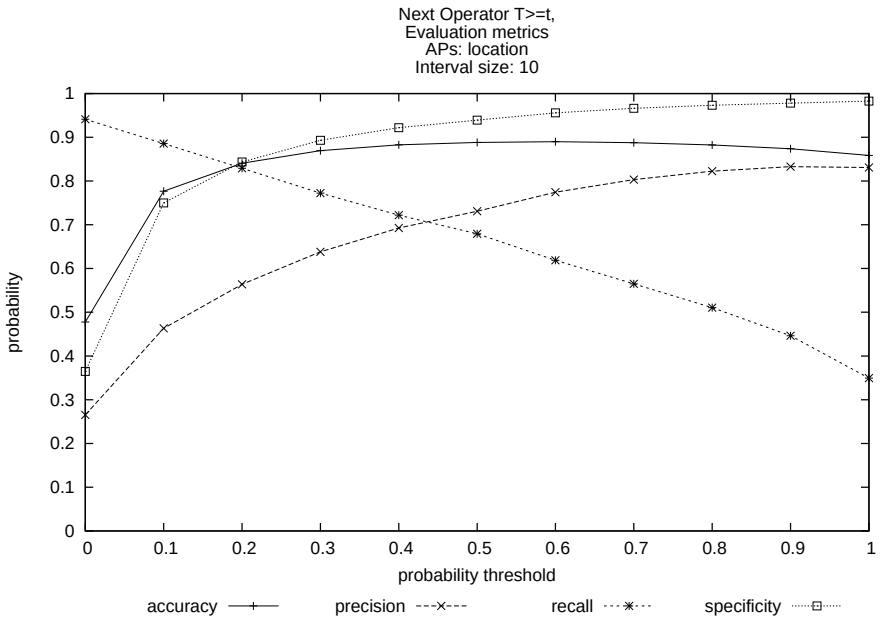

Figure 7.18: Evaluation of $X^{t \geq T}(\varphi)$ using Dartmouth data set

Figure 7.18 displays the evaluation results for all different metrics for the Dartmouth data set. In this data set accuracy stays around a value of 0.9 for $p \geq 0.3$. Precision already starts with a value of 0.3, with values of up to 0.75 for $p = 1.0$ and slowly rising to 1.0. At the same time recall drops with increasing values for the probability threshold p reaching 0.4 for $p = 1.0$. Specificity is higher than 0.8 from $p \geq 0.7$. The best setting for the

probability threshold p for all metrics here would be $p = 0.6$.

Conclusion In order to maximize the prediction results for the Mainkofen data set for all metrics we should therefore choose a value of $p = 0.5$ for the operator $X^{t \geq T}(\varphi)$. This ensures a high quality of predictions for all metrics with all different combinations of atomic propositions. Again the best performing combination of atomic properties in total is $location \times activity - concrete \times patient$.

In order to maximize the prediction results for the operator $X^{t \geq T}(\varphi)$ in the Dartmouth data set we should choose the probability threshold of $p = 0.6$.

We will use these values for the probability threshold for future comparisons of different operators when using $X^{t \geq T}(\varphi)$.

7.5.4 $\hat{X}^{t \geq T}$

This operator also takes the transition times on the DPDFs into account. This is achieved by offering a dwell time t that has already passed in the current state, from this point in time on we look into the future. For more details on this operator please refer to section 6.4.

The parameter t is passed in as a list of time stamps identifying the dwell times we want to evaluate for this operator. Again, if we switched contexts before the dwell time time stamp, i.e. $t_{real} < t$, we do not compute a result for this prediction but ignore it. We would also never compute this result because we would have already skipped to the follow up state s'.

Like before we compute $p = X^{t \geq T}(L(s''))$ for all successor states of s. The operator returns with the probabilities related to taking the specific transitions starting from time t. We now choose the state that is related to the highest probability we just computed with the regular $X^{t \geq T}(\varphi)$ operator. Now for all $q = \frac{k}{10}, k = 0, 1, 2, \ldots, 10$ we check if $p \geq q$, but this time again only for the state with the highest probability. If $p \geq q$ we predict the corresponding state to be the next state. If $p < q$ holds we predict the state is not going to be the next state. Now we compare our prediction with the actual next

state. This will again lead to a FP, TP, FN or TN. Furthermore, for each time window we save the number of FPs, TPs, FNs and TNs separately.

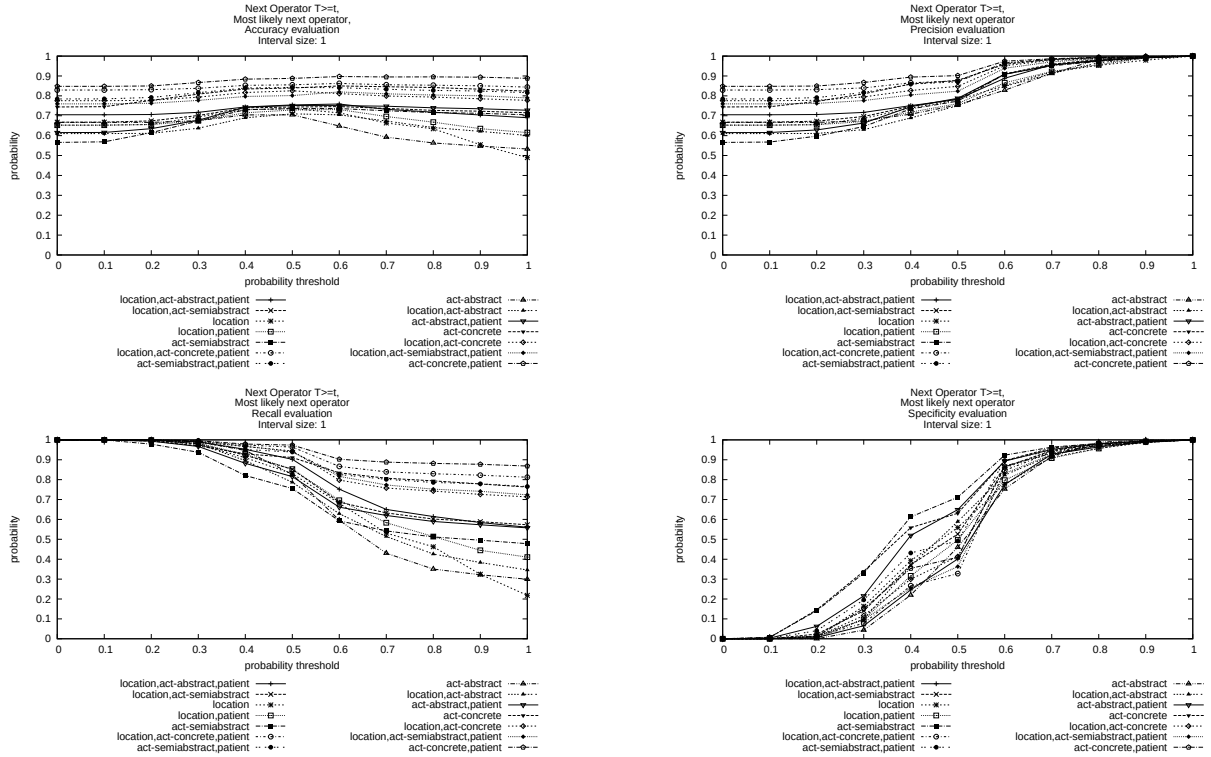

Figure 7.19: Metrics for $\hat{X}^{t \geq T}$ in the Mainkofen data set

Accuracy When taking a look at the evaluation results for accuracy in figure 7.19 we observe that almost all combinations of atomic propositions are almost constant and not dependant so much on the probability threshold p. For some combinations of atomic propositions like *location* or $location \times patient$ there is a global maximum around $p = 0.5$. In general we can say that all combinations of atomic propositions lie in between the interval $[0.5, 0.9]$.

The best performing combination of atomic propositions is $activity - concrete \times patient$ with a low of 0.85 and a high of 0.9, closely followed by $location \times activity - concrete \times patient$. The worst performing combination of atomic propositions is $activity - abstract$ with a low of 0.55 and a high of 0.65.

The best choice for the parameter probability threshold would be around $p = 0.5$ in order

to maximize the results for accuracy.

Precision Evaluating the results for precision we note that all combinations of atomic propositions lie in the interval $[0.55, 1.0]$.

The best performing combination of atomic propositions is $activity - concrete \times patient$ with a low of 0.85 and a high of 1.0, very closely followed by $location \times activity - concrete \times patient$. The worst performing combination of atomic propositions is $location \times activity - abstract$ with a low of 0.55 and a high of 1.0.

In order to maximize the results for precision the best choice for the parameter probability threshold would be for $p \geq 0.5$.

Recall All combinations of atomic propositions accomplish a ratio of higher than 0.2 with a maximum of 1.0. At probability threshold $p = 1.0$ the difference between the different combinations of atomic propositions is up to 0.7.

The atomic propositions with the best performance again are $activity - concrete \times patient$ with a low of 0.85, closely followed by $location \times activity - concrete \times patient$ with a low of 0.8. The combination of atomic propositions with the least values for recall again is $activity - abstract$.

The best choice for the probability threshold p here seems to be $p \leq 0.5$ in order to maximize the value for recall.

Specificity The diagram for specificity in figure 7.19 very much resembles the specificity diagram in figure 7.15 where we evaluated $\hat{X}$. Again we observe a huge increase of specificity when setting the probability threshold higher than 0.5. For $p \leq 0.5$ we have a maximum of 0.9 for all combinations of atomic propositions, but for $p \geq 0.6$ no combination of atomic propositions is less than 0.7. For $p \geq 0.6$ we get the best results overall with values higher than 0.7.

This time though the best performing combinations of atomic propositions is $activitiy - semiabstract$ with values over 0.9 for $p \geq 0.6$. The worst performing atomic proposition

is $activity - abstract$ with values of less than 0.75 for $p \leq 0.6$.

The best choice for the probability threshold p would be $p \geq 0.6$.

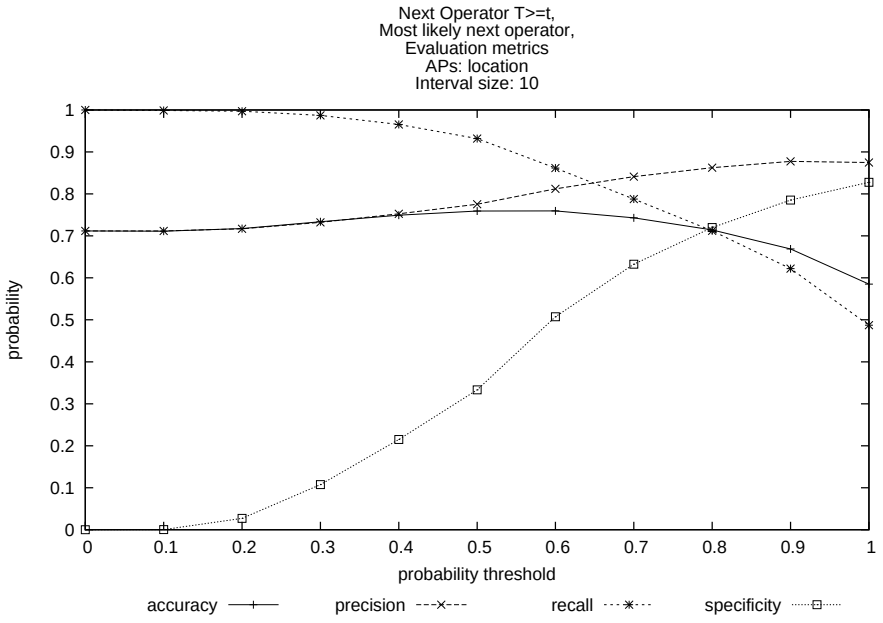

Figure 7.20: Evaluation of $\hat{X}^{t \geq T}$ using Dartmouth data set

Figure 7.20 displays the results for all metrics for the Dartmouth data set. Accuracy is very much constant at 0.7 for probability threshold values of $p \leq 0.9$, slightly dropping afterwards. Precision is rising from 0.7 at $p = 0.0$ up to 0.8 at $p = 1.0$. We notice that the recall value starts dropping very late with a relatively high value for the probability threshold $p \geq 0.5$ The same goes for specificity which is still at 0.3 at $p = 0.5$, rising up to 0.75. The best value for the probability threshold p here would be in the interval $[0.7, 0.8]$.

Conclusion For the Mainkofen data set the best performing combination of atomic propositions relatively clearly is $activity - concrete \times patient$. When choosing a value of $p = 0.6$ we can also make sure we get the best results overall. When looking at the Dartmouth data set we should choose a probability threshold of $p = 0.7$ in order to maximize all metrics. We will use these values for the probability threshold for future comparisons of different operators when using $\hat{X}^{t \geq T}$.

7.5.5 $X^{T \leq t \leq T+\Delta T}(\varphi)$

This operator also takes the transition times on the DPDFs into account. This time we are provided with a dwell time t and a time window $w = [t, t + \Delta t]$. For more details on this operator please refer to section 6.4.

If the context switch occurs before the dwell time parameter t, i.e. $t_{real} < t$ we again ignore the computation of the result and skip. If $t \leq t_{real} \leq t + \Delta t$ i.e. the real context switch occurs during the specified time window , we count the results in the following way: For each state $s'' \models \varphi''$ that the starting state s leads to we calculate $p = X^{T \leq t \leq T+\Delta T}(L(s''))$. Now for all $q = \frac{k}{10}, k = 0, 1, 2, \ldots, 10$ we check if $p \geq q$. If that is the case we predict the corresponding state to be the next state. If $p < q$ holds we predict the state is not going to be the next state. Now we compare our prediction with the actual next state. This will lead to a FP, TP, FN or TN. The parameter t and Δt is taken out of a list with different time windows. If $t_{real} > t + \Delta t$, i.e. the real context switch happens after the time window closes we have to count our results in a different way: If $p \geq q$ we evaluate the prediction as a FP since it may or may not be the next state, but it is definitely not happening during the time the time window specifies. Else, if $p < q$ we evaluate the prediction as a TN since no matter if the state is the correct follow up state, it is not following up during the time the time window specifies. For each time window we save the number of FPs, TPs, FNs and TNs separately.

Accuracy Looking at the accuracy diagram in figure 7.21 we see that the results for the different combinations of atomic propositions are extremely close together. All results are in the interval $[0.7, 0.8]$. We get the best overall results for values of the probability threshold p roughly for $p \geq 0.4$.

This time the combinations of atomic propositions *location* performs better than *location*$\times$ *activity* $-$ *concrete* $\times$ *patient* and *activity* $-$ *concrete* $\times$ *patient*, with values around 0.85 for $p \geq 0.3$. But in contrast to *location*, *location* $\times$ *activity* $-$ *concrete* $\times$ *patient* and *activity* $-$ *concrete* $\times$ *patient* perform better for small values of the probability threshold, $p \leq 0.1$. The worst performance is caused by the combination of atomic propositions *location* $\times$ *activity* $-$ *abstract* $\times$ *patient* with values roughly betweenabove 0.7 for $p \geq 0.4$. All other combinations of atomic propositions are in between.

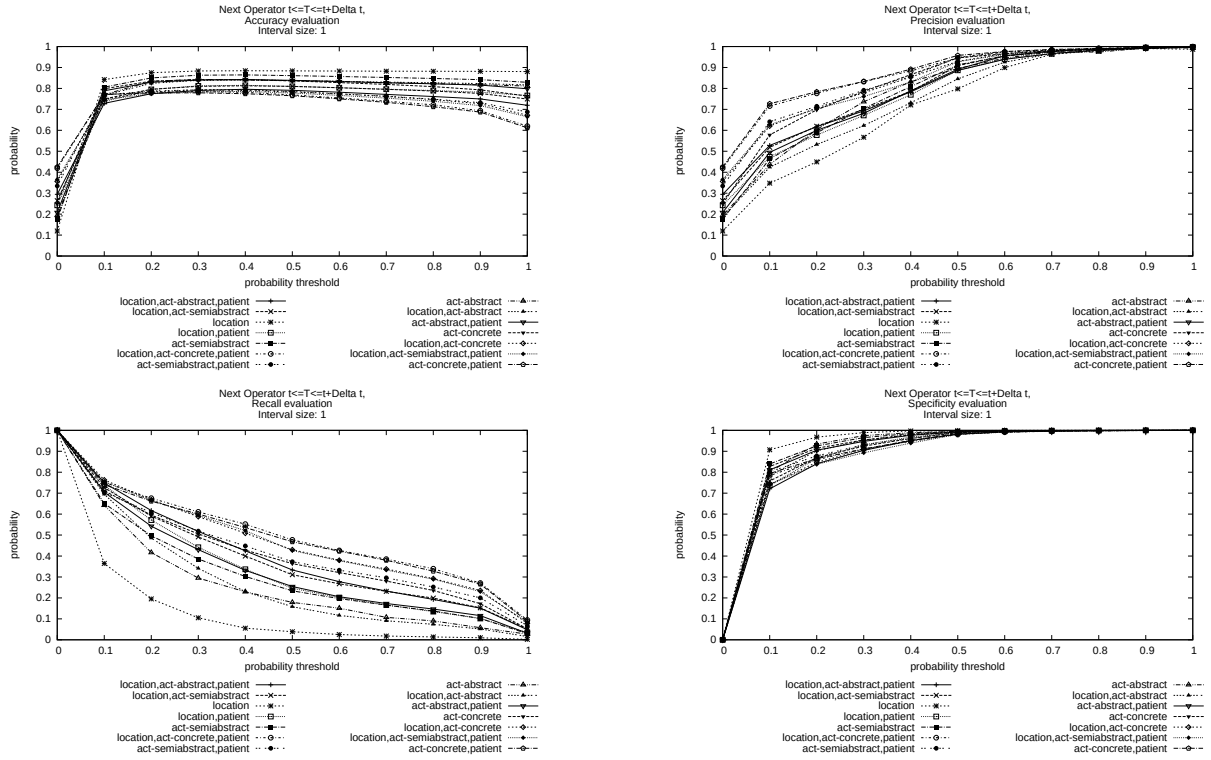

Figure 7.21: Metrics for $X^{T \leq t \leq T+\Delta T}(\varphi)$ in the Mainkofen data set

The best value for the probability threshold to choose here is $p \geq 0.4$.

Precision For the precision evaluation results we witness a different result. Starting from $p \geq 0.6$ the results for all combinations of atomic propositions are higher than 0.8.

The combination of atomic propositions $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best here with values between 0.5 and 1.0. Also again $activity - concrete \times patient$ performs slightly better than $location \times activity - concrete \times patient$. The atomic proposition $location$ performs worst, starting out at $p = 0.0$ with a precision value of 0.15. The best results are for the parameter probability threshold starting from $p \geq 0.6$.

Recall When looking at our evaluation metric recall we can observe that spread between different combinations of atomic propositions is big. The difference is as much as 0.7 for $p = 0.5$.

The combinations of atomic proposition $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best with values between 1.0 and 0.5. They perform about the same on average over all probability thresholds. The atomic proposition *location* clearly performs worst of all combinations again with values down to 0.15. Overall we get the best results for values p of the probability threshold about $p \leq 0.5$.

Specificity Examining specificity we observe most of the different combinations of atomic propositions are grouped together with a maximum difference of about 0.3. For $p \geq 0.6$ all results are grouped very tightly together.

Here the results are reversed again - the atomic proposition *location* scores the best results and $location \times activity - concrete \times patient$ scores worst. The best results are attained for $p \geq 0.5$.

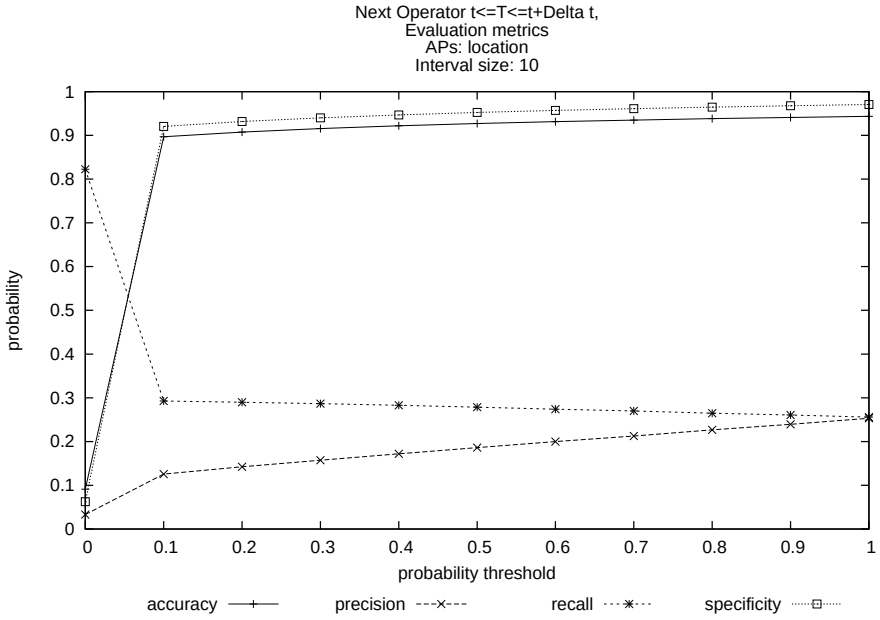

Figure 7.22: Evaluation of $X^{T \leq t \leq T+\Delta T}(\varphi)$ using Dartmouth data set

Taking a look at the evaluation results for the Dartmouth data we notice that accuracy, precision and specificity values are much lower compared to the Mainkofen evaluation results. In contrast to this the results for recall are very good especially for higher values of the probability threshold p. Accuracy goes up to 0.75 for $p = 1.0$ and in the interval of $[0.55, 0.75]$ for $p \geq 0.1$ in general. Precision is very low compared to the results from

the Mainkofen data set - its maximum is at about 0.42 with a value of more than 0.28 for $p \geq 0.1$. Recall always stays above 0.95 for all values of the probability threshold p which is a very good result. The curve for specificity looks exactly the same as the curve for accuracy but is slightly lower, having its maximum around $p = 1.0$ with a value of 0.72. Specificity is higher than 0.4 though for probability threshold values $p \geq 0.1$.

The best value to choose for the probability threshold in order to maximize all metrics is $p = 1.0$.

Conclusion In order to maximize the prediction results for the Mainkofen data set for all metrics we should therefore choose a value of $p = 0.5$ for the operator $X^{T \leq t \leq T+\Delta T}(\varphi)$. This will ensure the highest possible quality of predictions for all metrics with all different combinations of atomic propositions. With this operator is not so clear which the best performing combination of atomic propositions is. Performing about the same, we decide to go for $activity - concrete \times patient$ instead of *location* since all other operators so far already use this parameter.

In order to maximize the prediction results for the operator $X^{T \leq t \leq T+\Delta T}(\varphi)$ in the Dartmouth data set we we decide to use a probability threshold of $p = 1.0$.

We will use these values for the probability threshold for future comparisons of different operators when using $X^{t \geq T}(\varphi)$.

7.5.6 $\hat{X}^{T \leq t \leq T+\Delta T}$

This operator also takes the transition times on the DPDFs into account. This time we are provided with a dwell time t and a time window $w = [t, t + \Delta t]$. For more details on this operator please refer to section 6.4.

If the context switch occurs before the dwell time parameter t, i.e. $t_{real} < t$ we again ignore the computation of the result and skip. If $t \leq t_{real} \leq t + \Delta t$ i.e. the real context switch occurs during the specified time window , we count the results in the following way: Like before we compute $p = X^{T \leq t \leq T+\Delta T}(L(s''))$ for all successor states of s. The operator returns with the probabilities related to taking the specific transitions starting from time

t during a time window of $w = [t, t + \Delta t]$. We now choose the state that is related to the highest probability we just computed with the regular $X^{T \leq t \leq T+\Delta T}(\varphi)$ operator. Now for all $q = \frac{k}{10}, k = 0, 1, 2, \ldots, 10$ we check if $p \geq q$, but this time again only for the state with the highest probability. If $p \geq q$ we predict the corresponding state to be the next state. If $p < q$ holds we predict the state is not going to be the next state. Now we compare our prediction with the actual next state. This will again lead to a FP, TP, FN or TN. If $t_{real} > t + \Delta t$, i.e. the real context switch happens after the time window closes we have to count our results in a different way: If $p \geq q$ we evaluate the prediction as a FP since it may or may not be the next state, but it is definitely not happening during the time the time window specifies. Else, if $p < q$ we have to evaluate the prediction as a TN since no matter if the state is the correct follow up state, it is not following up during the time the time window specifies. For each time window we save the number of FPs, TPs, FNs and TNs separately. Furthermore, for each time window we save the number of FPs, TPs, FNs and TNs separately.

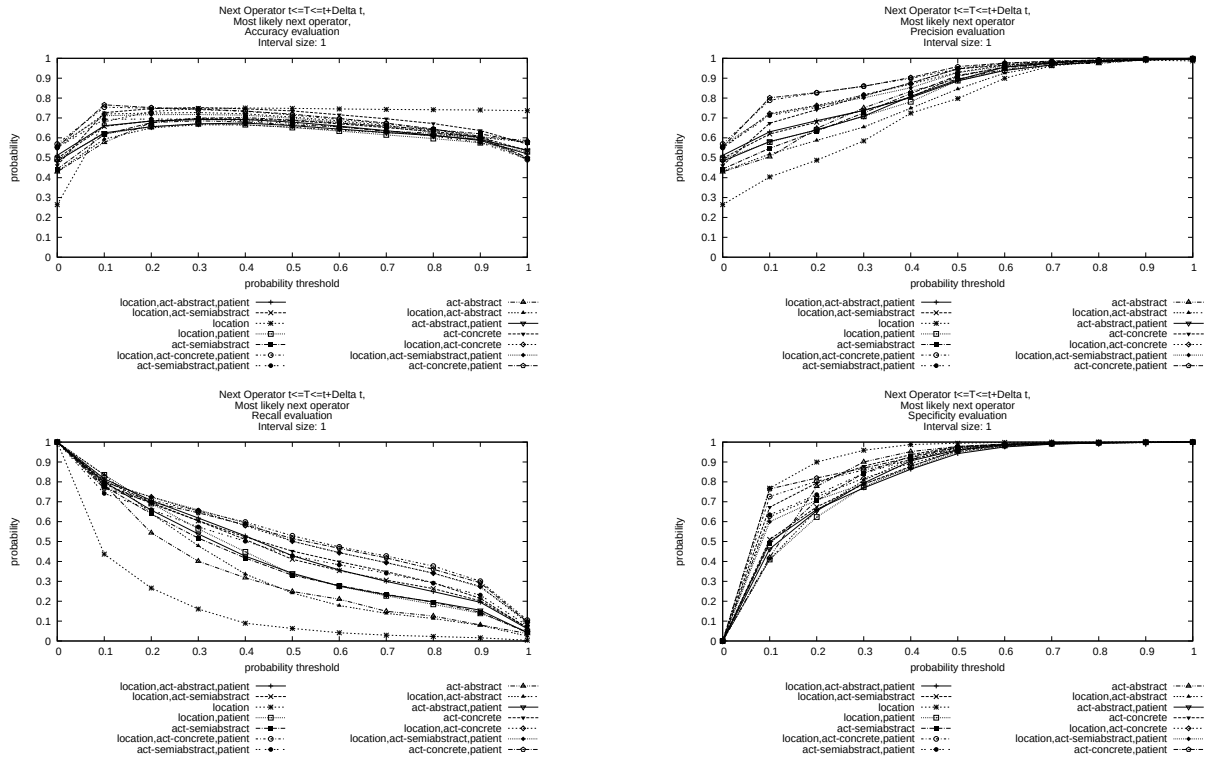

Figure 7.23: Metrics for $\hat{X}^{T \leq t \leq T+\Delta T}$ in the Mainkofen data set

Accuracy Looking at the accuracy diagram in figure 7.23 we see that the results for the different combinations of atomic propositions all show the same curve, just the offsets are different. All results are in the interval $[0.4, 0.9]$. We get the best overall results for values of the probability threshold p roughly for $0.4 \leq p \leq 0.6$.

The combinations of atomic propositions $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best, with values around 0.8 to 0.85 for $p \leq 0.9$. The atomic proposition *location* performs worst of all with values between 0.45 and 0.6. All other combinations of atomic propositions are in between.

In order to maximize accuracy the best value for the probability threshold to choose here is $0.4 \leq p \leq 0.6$.

Precision When examining figure 7.23 for precision results we witness a similar result. Starting from $p \geq 0.6$ the results for all combinations of atomic propositions are higher than 0.8.

The combination of atomic propositions $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ again perform best with values between 0.85 and 1.0. Again $activity - concrete \times patient$ performs slightly better than $location \times activity - concrete \times patient$. The atomic proposition *location* performs worst, starting out at $p = 0.0$ with a precision value of 0.45. The best results are for the parameter probability threshold starting from $p \geq 0.6$.

Recall Examining our evaluation metric recall we can observe that spread between different combinations of atomic propositions is big. The difference is as much as 0.75 for $p = 0.9$.

The combinations of atomic proposition $location \times activity - concrete \times patient$ and $activity - concrete \times patient$ perform best with values between 1.0 and 0.5. The combination of atomic propositions $activity - concrete \times patient$ performs slightly better though. The atomic proposition *location* clearly performs worst of all combinations again with values down to 0.05. Overall we get the best results for values p of the probability threshold about $p \leq 0.4$.

Specificity Evaluating the results for specificity we can again witness a big jump for all combinations of atomic propositions from $p = 0.5$ to $p = 0.6$. For $p \geq 0.6$ all results are grouped very tightly together.

Here the results are exactly reversed - the atomic proposition *location* scores the best results and $location \times activity - concrete \times patient$ scores worst. The best results are attained for $p \geq 0.5$.

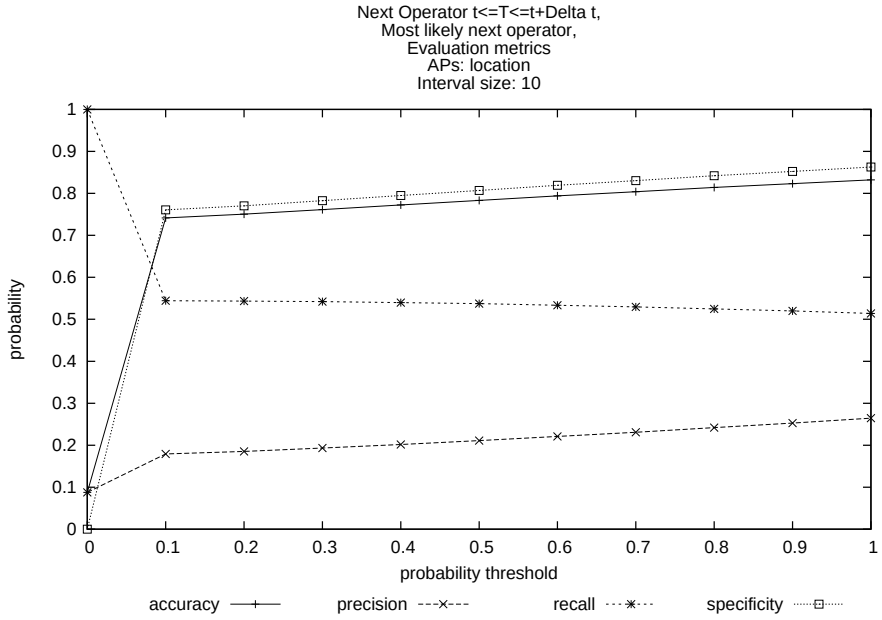

Figure 7.24: Evaluation of $\hat{X}^{T \leq t \leq T+\Delta T}$ using Dartmouth data set

Taking a look at the evaluation results for the we notice that accuracy, precision and specificity values are almost constant. Accuracy is at a value of about 0.75 for all values for the probability threshold p. Precision behave very much the same way as accuracy does with a value of roughly 0.75 too. Recall is at almost 1.0 throughout all values of the probability threshold p. Specificity though is almost at 0 for all values of p. When looking at the absolute numbers we can see that there are hardly any FNs or TNs at all. This means that almost always the values for the probability threshold are below the result of the operator, resulting only in TPs or FPs.

The best value to choose for the probability threshold in order to maximize all metrics is $p = 1.0$.

Conclusion In order to maximize the prediction results for the Mainkofen data set for all metrics we should therefore choose a value of $p = 0.5$ for the operator $\hat{X}^{T \leq t \leq T+\Delta T}$. This will ensure the highest possible quality of predictions for all metrics with all different combinations of atomic propositions. With this operator is not so clear which the best performing combination of atomic propositions is. Performing about the same, we decide to go for $activity - concrete \times patient$ instead of *location* since all other operators so far already use this parameter.

In order to maximize the prediction results for the operator $\hat{X}^{T \leq t \leq T+\Delta T}$ in the Dartmouth data set we we decide to use a probability threshold of $p = 1.0$.

We will use these values for the probability threshold for future comparisons of different operators when using $\hat{X}^{T \leq t \leq T+\Delta T}$.

7.5.7 $X^{-1}(\varphi)$

This operator also takes time into account. For more details on this operator please refer to section 6.4.

In order to evaluate this operator we compute $t_{predicted} = X^{-1}(p, L(s'))$ for all $q = \frac{k}{10}, k = 0, 1, 2, \ldots, 10$. If the context switch t_{real} occurs in the predicted time interval, i.e. $t_{real} \in [0, t_{predicted}]$ we count the result as a TP. If it is outside this interval we count the result as a FP. If the probability can not be reached at all the operator returns $false$. This can happen if a context is reachable only via a very low transition probability. If the result is $false$ we do not count it. We save the absolute error $abs(t_{real} - t_{predicted})$ in order to reason about the average absolute errors. For each q we save the number of FPs, TPs and absolute errors separately.

Absolute errors The diagram on the left in figure 7.25 shows the average absolute error for all probability thresholds $q = \frac{k}{10}, k = 0, 1, 2, \ldots, 10$ for the Mainkofen data set. The absolute errors are surprisingly low and are for the most part for almost all combinations of atomic propositions in a less than 5 seconds window. The only exception to this is the atomic proposition *location* of up to 30 seconds of average absolute error at $p = 0.1$. The best performing combination of atomic propositions is $activity - abstract \times patient$,

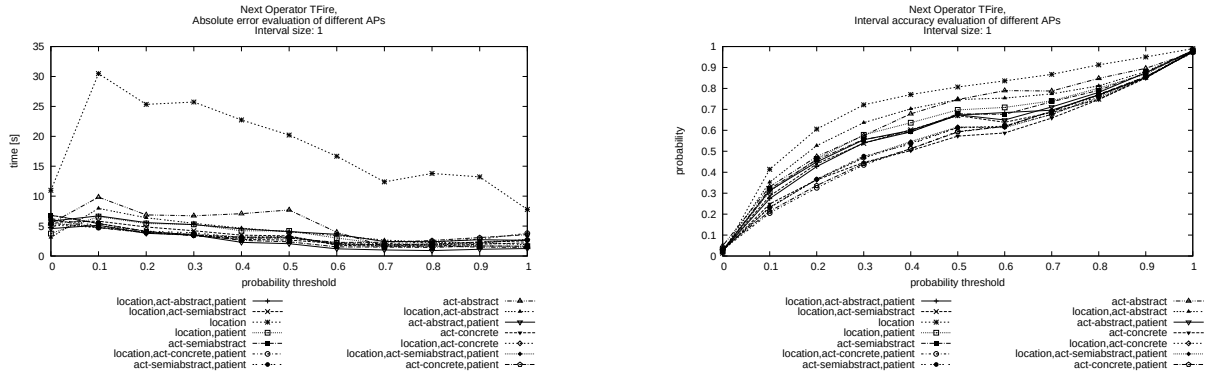

Figure 7.25: Metrics for $X^{-1}(\varphi)$ in the Mainkofen data set

although all combinations are very tightly grouped together. We can estimate that the higher the value for the probability threshold p the better our results get. So we get the best results for $p = 1.0$.

Interval Precision The diagram on the right in figure 7.25 displays the precision metrics for the Mainkofen data set. The results show that with an increasing probability threshold p the precision value gets better and better, with values over 0.8 starting from $p \geq 0.9$. The best performing combination of atomic propositions here clearly is *location*. The worst performing combination is $activity - concrete$.

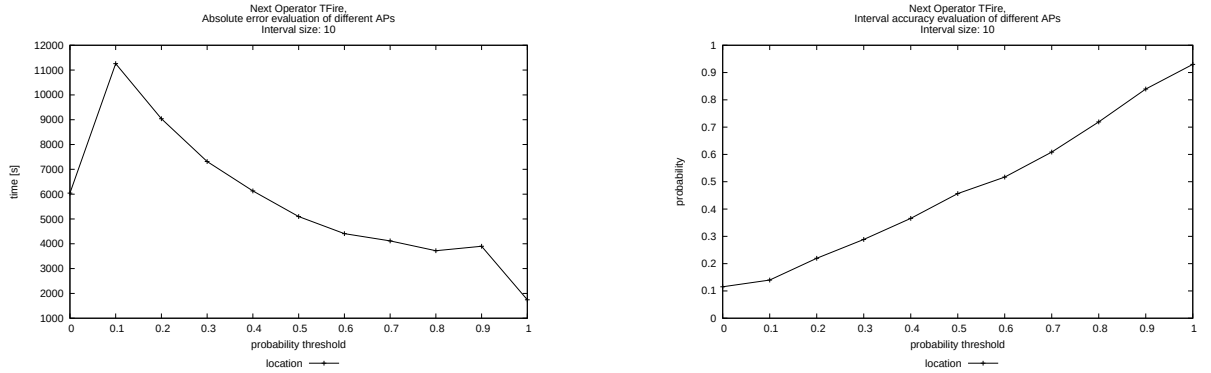

Figure 7.26: Absolute error evaluation of $X^{-1}(\varphi)$ using Dartmouth data set

Figure 7.26 examines the Dartmouth data set results for the operator $X^{-1}(\varphi)$. For the atomic proposition *location* we get very good results for the interval precision, that is if

the prediction time window actually included the real context switch timestamp. At the same time though the actual absolute error for *location* is pretty high.

The best trade off here would be to choose $location \times patient$ with a probability threshold of $p = 1.0$. Examining the left diagram in figure 7.26 we can again establish that a higher value for the probability threshold p will result in less average absolute error. The absolute error reaches its maximum at $p = 0.1$ with about 3 hours of absolute error and goes down to 30 minutes at $p = 1.0$. The best value for the probability threshold p here would be $p = 1.0$. The interval accuracy in figure 7.26 also rises very fast from about 0.15 at $p = 0.0$ to 0.9 at $p = 1.0$. The best value here for the probability threshold p again would be $p = 1.0$.

Conclusion The reason for the worse performance of the operator in the Dartmouth data set compared to the Mainkofen data set is that the spread of timestamps is much wider for the Dartmouth data set. As we already established about 60% of all context switches here occur within the first 40 seconds, but the remaining 40% of all context switches occur at some later time, for some context switches many hours later. Even after 60 seconds not more than 65% of all context switches have taken place. In contrast to this with the Mainkofen data set more than 85% of all context switches occur within the first 40 seconds. After 60 seconds more than 95% of all context switches have occured.

In order to optimize both metrics for the Dartmouth data set we should choose a probability threshold of $p = 1.0$. For the Mainkofen data set a probability threshold value of $p = 1.0$ should be chosen as well.

7.5.8 $\Phi U^{T \leq t \leq T+\Delta T} \Psi$

This operator also takes the transition times on the DPDFs into account. This time we are provided with a time t and a time window $w = [t, t + \Delta t]$. For more details on this operator please refer to section 6.4.

If the context switch occurs before the dwell time parameter t, i.e. $t_{real} < t$ we again ignore the computation of the result and skip. Starting in state s we perform a depth first search (DFS) through the transition system, recording all locations occurring on the

way. For each new location we encounter we calculate $p = trueU^{T \leq t \leq T+\Delta T}\Psi, \varphi = L(s'')$. If we hit a location property we already recorded the result for we can safely ignore it, since we already computed exactly the same query with a previous computation. The DFS stops either when a time threshold is passed (for example for the Mainkofen data set we used a threshold of 30s) on each path or when the probability along the path falls below a minimum probability.

Now we choose the location property with the highest probability p for our evaluation. Now for all $q = \frac{k}{10}, k = 0, 1, 2, \ldots, 10$ we check if $p \geq q$. If that is the case we predict the corresponding location property to occur within the time window $w = [t, t + \Delta t]$. If $p < q$ holds we predict the location property is not going to occur. Now we compare our prediction with the actual next location properties in the random user trace up to time $t + \Delta t$. This will lead to a FP, TP, FN or TN. The parameter t and Δt is taken out of a list with different time windows. If $t_{real} > t + \Delta t$, i.e. the real context switch happens after the time window closes we have to count our results in a different way: If $p \geq q$ we evaluate the prediction as a FP since it may or may not be the next state, but it is definitely not happening during the time the time window specifies. Else, if $p < q$ we evaluate the prediction as a TN since no matter if the state is the correct follow up state, it is not following up during the time the time window specifies. For each time window we save the number of FPs, TPs, FNs and TNs separately.

Since the evaluation process consists of another depth first search the complexity of the evaluation for this operator is quite high and is mostly bounded by the size of the time window we are interested in: The complexity of performing this evaluation is in $O(l * n^k)$, with l being the number of different locations we can encounter during the time window, n being the average degree of outgoing transitions of a state in the transition system and k being the depth of the search, which is limited by the time window.

Therefore we could only evaluate time windows of relatively small sizes. For the Mainkofen data set we chose a time window of 10 seconds for the prediction. Because of the large size of the Dartmouth data set we had to go even lower in order to compute the results and went down to 5 seconds.

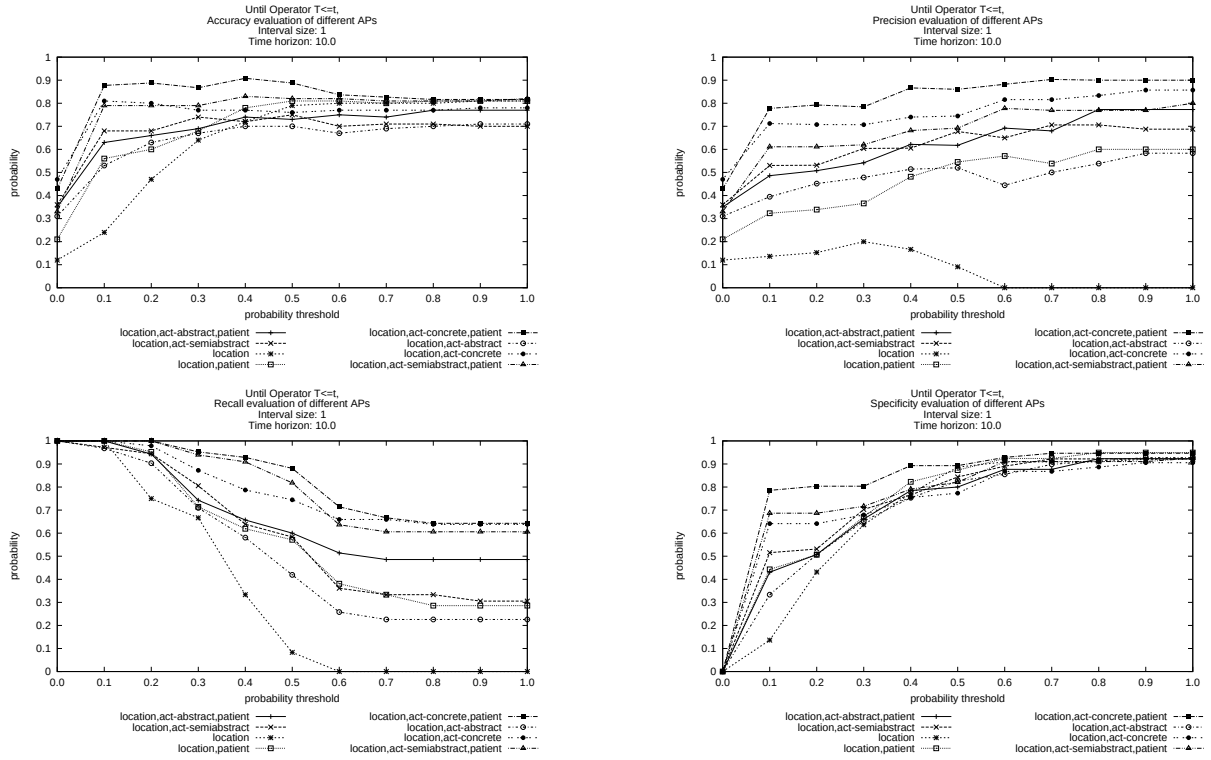

Figure 7.27: Metrics for $\Phi U^{T \leq t \leq T+\Delta T} \Psi$ in the Mainkofen data set

Accuracy Starting from probability thresholds $p \geq 0.3$ all accuracy results for all combinations of atomic propositions are higher than 0.6 when looking at figure 7.27. The best performing combination of atomic propositions is $location \times activity - concrete \times patient$ with accuracy over 0.8 for $p \geq 0.1$. The worst performing combination of atomic propositions is $location \times activity - abstract$ with an accuracy between 0.6 and 0.7 for $p \geq 0.3$.

Precision For the precision results again $location \times activity - concrete \times patient$ performs best with values of greater than 0.8 for probability thresholds $p \geq 0.4$. The atomic proposition location alone clearly performs worst here with of less than 0.2 for $0.0 \leq p \leq 0.6$. The reason behind this is that the number of TPs falls down to 0 after this threshold. Again there are two reasons behind this: One definitely is the very limited time window supplied, the other is the properties of the Dartmouth data set, having almost no transitions with transition probabilities of higher than 0.5.

Recall Evaluating the recall results the combination of atomic propositions *location* × *activity* − *concrete* × *patient* again performs best with a difference of up to 0.6 to the worst performing atomic proposition *location*. Again this is caused by a number of 0 TPs from $p \geq 0.6$.

Specificity All combinations of atomic propositions perform rather well and result in a specificity value of between 0.7 and 0.95 for probability thresholds $p \geq 0.4$. Again *location* × *activity* − *concrete* × *patient* performs best.

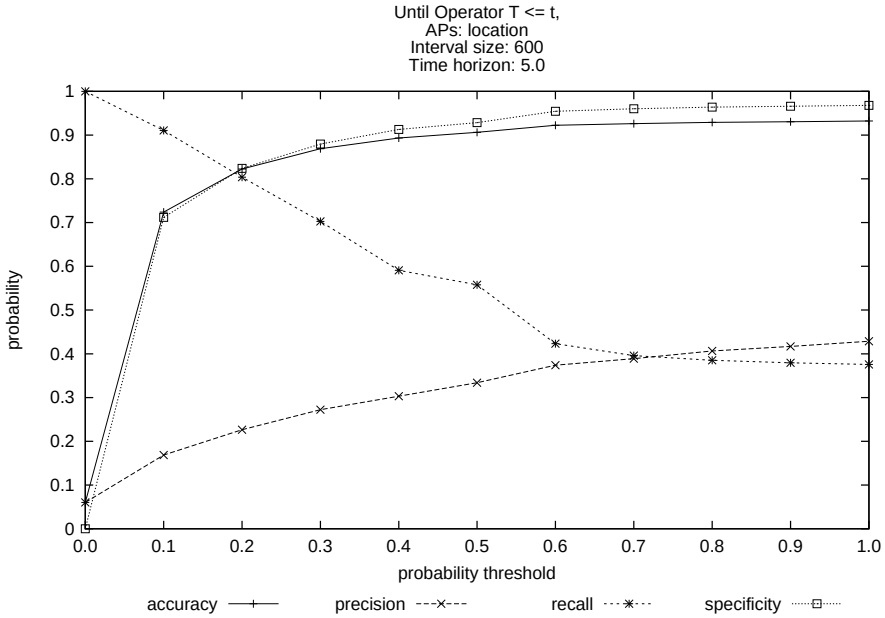

Figure 7.28: Metrics for $\Phi U^{T \leq t \leq T+\Delta T} \Psi$ in the Dartmouth data set

Figure 7.28 shows the evaluation results for the Dartmouth data set. Accuracy is higher than 0.7 starting from $p \geq 0.1$ and is mostly in in the interval $[0.8, 0.9]$ for higher values of the probability threshold. Precision is low, it never is higher than 0.4 and is about 0.3 on average over all values for the probability threshold. This is caused by more FPs than TPs, which can easily be explained by the small time window size. The way we evaluate the operator we will always assume one context to be a future context and given only a small time window size this context does not necessarily reflect the real next context. Recall goes down to 0.4 starting from $p \geq 0.6$ and specificity behaves very much like accuracy.

The best value for the probability threshold would be $p = 0.6$ in order to maximize the results for all metrics. Using a bigger time window would very likely increase the precision results a lot.

Conclusion The best value for the probability threshold would be $p = 0.6$ in order to maximize the results for all metrics for both data sets. Using a bigger time window would very likely increase the precision results a lot.

An interesting side note: Using only the location property for $\Phi U^{T \leq t \leq T+\Delta T} \Psi$ performs with significantly worse precision and recall results than when incorporating more information like *patient* or $activity - concrete$.

7.6 Results

7.6.1 Effects of probability thresholds

All operators we supply in this thesis come with a probability threshold p that controls if the context predictor will report a positive or a negative prediction. I.e. when the numeric result internally computed by the context predictor fulfills $P \cong p$ the result the context predictor will return will be true, otherwise it will be false.

Since the user has to supply the parameter in order to get predictions we examined in the previous section how this probability threshold affects the prediction results. The results of our evaluation show a trade off scenario between the different metrics we compared. That is why in general we can not give a good number to choose for the probability threshold p - it depends on the user traces and the Semi-Markov chains generated by these user traces. Also it depends a lot on the application querying the context predictor and the requirements of this application. We can generally try to answer the question for the Mainkofen and Dartmouth data sets though.

Increasing the probability threshold value p generally always results in higher precision results, i.e. a higher rate of TPs in comparison of FPs. This is of course a feature that is quite welcome since it means that if the predictor predicts a context is coming up next the probability of happening so rises with increasing p.

Also when increasing the probability threshold accuracy rises up to a point, generally about $p \leq 0.7$ The reason it is not monotonously rising further is that a higher probability threshold can lead to a high number of FNs, since a lot of transitions already have transition probabilities below the threshold and are therefore being predicted as not following up next as a context. Generally a high accuracy value is welcome, since it compares the general true predictions the context predictor made with all predictions made.

When looking at specificity we get similar results again, increasing the probability threshold leads to a higher specificity value. A high specificity is also very welcome since it means we don't have many FPs in comparison to the number of TNs. The result here makes sense, increasing the probability threshold will lead to an increase in TNs since for

most transitions being below the treshold will mean they actually are not going to be the next active context. At the same time increasing the probability threshold will decrease the number of FPs since if a transition already has a high transition probability it is in reality very very likely to lead to the next active context.

Until here it looks like having a relatively high value for the probability threshold will result in very good results. Now comes the tradeoff: recall is generally always decreasing with a rising probability threshold p. This means we have a tradeoff in a higher rate of FNs in comparison with TPs when increasing the probability threshold p.

Therefore to get an overall good result we suggest to set the probability threshold to values $0.4 \leq p \leq 0.7$ which will maximize all metrics. If the application requires a high recall result though we suggest to decrease the probability threshold to lower values resulting in worse results for the other metrics. If recall is not as important we can further increase p and gain better results for the other metrics.

7.6.2 Effects of time window sizes

The time window size $t + \Delta t$ has a high impact especially on $X^{T \leq t \leq T+\Delta T}(\varphi)$. Setting the time window size to a very big number, i.e. representing an "unlimited" value we get the same results as when using the operator $X^{T \geq t}(\varphi)$. This makes a lot of sense since $X^{T \leq t \leq T+\Delta T}(\varphi)$ is a generalization of $X^{T \geq t}(\varphi)$.

Most of the time an application will be interested in choosing a smaller time window though to perform predictions that are precise in this very time window. Of course this is only possible if the transition is highly dependant on context switch times and not so much on transition probabilities, and also it requires the correct time window sizes. I.e. if the time window is too small we will get bad results especially for recall. This is because the context predictor will always report a context will not show up next if $t_{real} \notin [t, t + \Delta t]$, i.e. the true context switch time is not in the time window $t + \Delta t$.

Of course the same applies to $\hat{X}^{T \leq t \leq T+\Delta T}(\varphi)$.

7.6.3 Necessary granularity for time intervals

In general the finer the granularity for the discrete probability density functions the better our results will be. We experimented with different interval sizes Δt and came up with the conclusion that we get the best results for the smallest possible size. But the interval size Δt again leads to a tradeoff in prediction quality versus memory size and CPU power requirements. The Dartmouth data set for example comes with over 9 million time stamps over all user traces. In general there are lots of transition systems generated by the Dartmouth user traces that emply hundreds of thousands of time stamps for context switches. Since we already showed that the time stamps are spread over a very wide area of time this means those time stamps will also consume a large amount of memory. Also when computing the operators we need to integrate over the discrete probability density functions, meaning we have to perform more steps if the amount of intervals is higher because the interval size Δt is lower. Therefore for the Dartmouth data set we had to choose an interval size of $\Delta t = 10$ seconds in order to make an evaluation possible at all. We generally advise to set the interval size Δt to the lowest value possible that still performs well in terms of memory and CPU power.

7.6.4 Semi-Markov chains vs continuous time Markov chains

Semi-Markov chains model a "memoryless" negative exponential probability density distribution. This means that in continuous time Markov chains it does not make any difference how much time passes by in a state for a context switch to occur to a different context. We believe that this does not always accurately model reality. In fact, a simple example illustrates the opposite: When entering a taxi and staying for a long time in this taxi the probability of going to a far away location rises with the time spent being in the taxi. So the more time is spent in the context of being in the taxi, the more likely it is for the next context to be a further away location in general, assuming the taxi driver takes the shortest paths. This property can not be modeled by continuous time Markov chains.

The Meinkhoven data set again shows that in reality often behaviour is not governed by a negative exponential probability density distribution. Due to the lack of amount of user traces we had to generate data for the transitions' discrete probability density functions.

We modeled those time stamps to follow a Gaussian distribution. With our custom built discrete probability density functions we were able to approximate the behaviour very closely and to gain good results with our metrics.

For the Dartmouth data set many transitions display a behaviour very closely resembling a negative exponential probability density distribution. Of course in this setting we do not gain any benefits by using Semi-Markov chains instead of using continuous time Markov chains. But in case a data set does not follow a negative exponential probability density distribution we can closely model that behaviour with our approach of using Semi-Markov chains.

It still remains an open task to find a suitable real world data set that proves that Semi-Markov chains are superior to continuous time Markov chains in a real world example.

7.6.5 Semi-Markov chains vs discrete time Markov chains

In this section we compare the prediction quality of the different next operators we proposed especially in contrast to Model Checking discrete time Markov chains which only takes the transition probabilites into account and not the time that has been spent in a context. In order to narrow the parameters down to a feasible size we use the findings from section 7.5. Since in most cases the combination of atomic propositions $activity - concrete \times patient$ performed best we will use it here to compare the different operators.

We start by comparing the Next operators for the Mainkofen and Dartmouth data sets followed by the most likely Next operators again for both data sets. Again we compare accuracy, precision, recall and specificity.

We start by evaluating the results for the next operators displayed for the Mainkofen data set in figure 7.29.

Accuracy The operator $X^{t \geq T}(\varphi)$ performs slightly better than the operator $X(\varphi)$ over all probability thresholds p, suggesting that in fact there is a difference if taking more than only transition probabilities into account. The advantage here is only little though

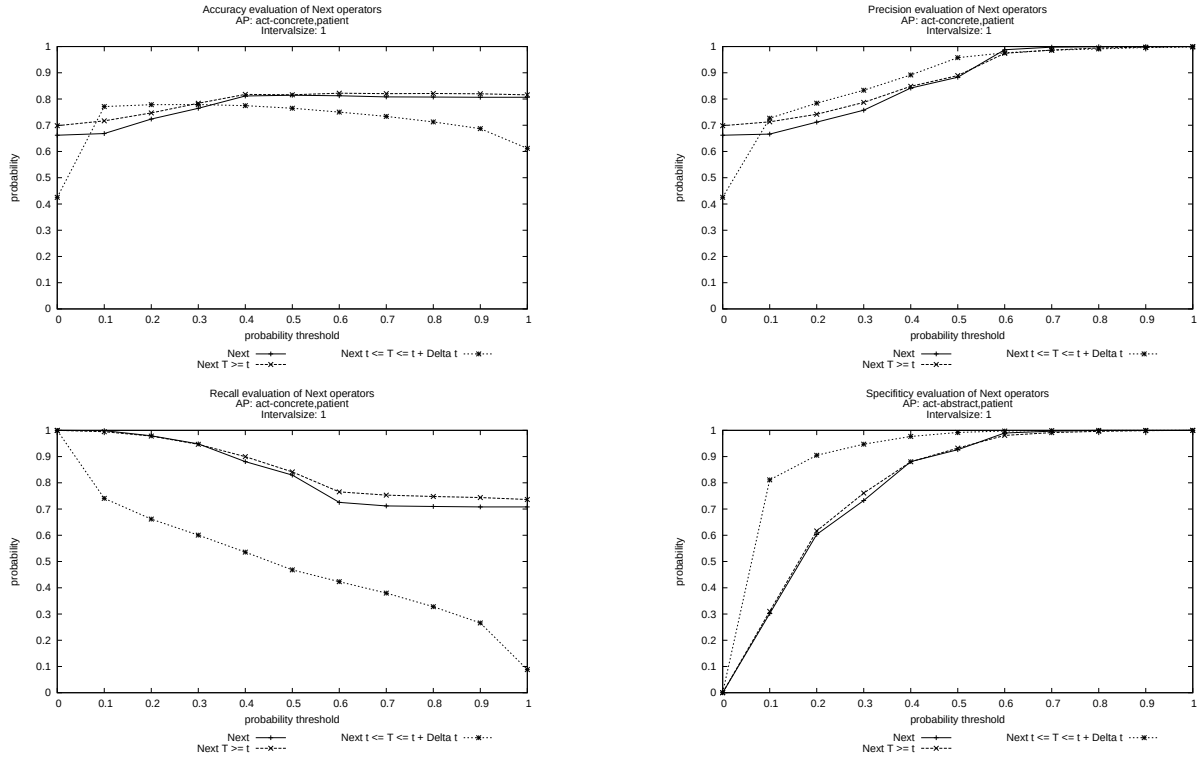

Figure 7.29: Operator prediction quality comparison for Next operators in the Mainkofen data set

and for the most part the difference is not more than 5%. In comparison $X^{T \leq t \leq T+\Delta T}(\varphi)$ performs far worse - this is due to the fact that the operator takes only a small time window into account and therefore always has a very limited prediction window in comparison to $X^{t \geq T}(\varphi)$. Also, if we predict a FP because a context switch is occuring outside of the time window we still predict a FP, even though it might be the correct context. The operator performs about 10% to 15% worse than the other two operators. For small probability thresholds $p \leq 0.3$ the operator performs better than the other two operators. The reason for this is that transitions with low probabilities are taken into account here. So in fact this operator becomes useful when having transition systems with a higher average degree of outgoing transitions.

Precision Precision is higher than 0.8 starting from $p \geq 0.4$ for all operators. Precision is not as good for $X(\varphi)$ for probability threshold values of $p \leq 0.6$ due to the fact that transitions with lower transition probabilities add more unconfidence and the operator

does not take time into account. The difference to the operators taking time into account here is more than 10%. Still the results are very high even for $X(\varphi)$. The best performing operator in terms of precision here is $X^{T\leq t\leq T+\Delta T}(\varphi)$ for lower probability thresholds ≤ 0.6. This can again be explained by the fact that the operator does take time into account and the number of TPs can be increased by adding the time window knowledge. $X^{t\geq T}(\varphi)$ performs in between the other operators between $p0.1 \leq 0.4$, this suggests that for smaller values of the probability threshold it is worth using a time window instead of only taking the time already spent in the state into account or only the transition probabilities.

Recall $X^{T\leq t\leq T+\Delta T}(\varphi)$ clearly performs worst here. This is a lot due to the fact that the number of FN is very high. This is caused by recording a FN when the predicted time window did not include the real time of the context switch, even though the future context would have been correctly identified as being the next. This is confirmed when looking at the recall results of $X^{t\geq T}(\varphi)$. it performs better than $X(\varphi)$ especially for higher values of the probability threshold $p \geq 0.6$. So even though some transitions have relatively high transition probabilities it still is possible to take a different transition with a lower probability depending on the time already spent in the current context. The difference between $X(\varphi)$ and $X^{t\geq T}(\varphi)$ is about 5%.

Specificity Here $X^{T\leq t\leq T+\Delta T}(\varphi)$ clearly outperforms the other two operators by far. The reason behind this again is the time window we provide the operator with - there are hardly any FPs generated in comparison with TNs since the time window is a very precise instrument for sorting out future contexts. The number of TNs for this operator is very high. The other two operators behave very much the same, suggesting that for specificity there is not much of a difference no matter if using $X(\varphi)$ or $X^{t\geq T}(\varphi)$.

Conclusion When looking at the evaluation results for the Mainkofen data set we can conclude the following facts. $X^{T\leq t\leq T+\Delta T}(\varphi)$ is especially good for transition systems with high degrees of outgoing transitions. In this case, when a high specificity or a high precision is needed one should use this operator. The drawback is the low recall results for this operator.

$X^{t \geq T}(\varphi)$ generally also performs better than $X(\varphi)$ and therefore is first choice before falling back to using $X(\varphi)$. Compared to $X^{T \leq t \leq T+\Delta T}(\varphi)$ the recall results are far better, by up to 100%.

In fact the Mainkofen data set reflects our originary assumptions for the three operators. Time is definitely a factor that has to be taken into account in order to predict future contexts with a higher quality. The difference is not very high but taking into account that we had to generate most of the context switch times it is possible that using a different real world data set the results might even be better.

Especially looking only at a time window can have a big impact on precision and specificity, but also on recall.

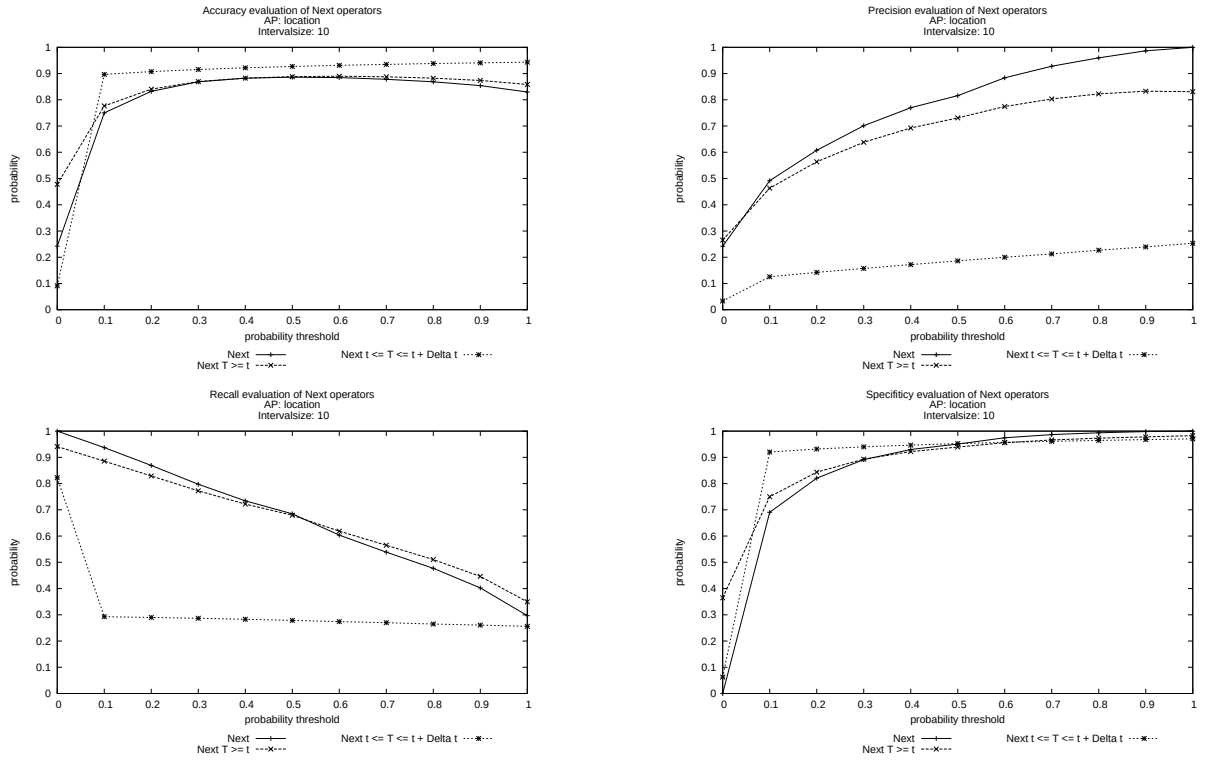

Figure 7.30: Operator prediction quality comparison for Next operators in the Dartmouth data set

We now evaluate the results for the next operators displayed for the Dartmouth data set in figure da7.30.

Accuracy In the Dartmouth data set the accuracy results are different. Clearly $X^{T\leq t\leq T+\Delta T}(\varphi)$ delivers the best accuracy results with slow linear growth with increasing values for the probability threshold. The reason for the almost perfect linear growth is the intervall size Δt. Because of the large amount of data and context switches provided in the data set it was not feasible to use a better resolution for the time intervals. Therefore the results are less fine grained. This is especially true for $X^{T\leq t\leq T+\Delta T}(\varphi)$ since with this operator we possibly truncate time intervals from two sides, depending on the parameters t and $t+\delta t$. This holds true for the other metrics as well. The results are clearly better than the for the other two operators though by about 10%. $X^{t\geq T}(\varphi)$ performs slightly better than $X(\varphi)$ but not in a significant way.

Precision When looking at the precision results $X(\varphi)$ clearly performs best. The difference in comparison with the other operators is as much as 0.75 for $X^{T\leq t\leq T+\Delta T}(\varphi)$ and 0.2 for $X^{t\geq T}(\varphi)$. $X^{t\geq T}(\varphi)$ still has good results with precision results higher than 0.7 for probability threshold values $p \geq 0.6$. This means that in comparison to $X(\varphi)$ the other two operators have a very high rate of FPs in contrast to TPs. Since the transition times are spread over a wide area of time this result is not so surprising anymore - when only looking at a relatively small (in this case 2 hours) time window only a fraction of less than 70% of all context switch times can be observed. With an unlimited time window $X^{T\leq t\leq T+\Delta T}(\varphi)$ would behave exactly like $X^{t\geq T}(\varphi)$, so the results here are more questioning the size of the time window. Two hours seemed to be a good estimate though for the average duration of stay at a location at a university campus. Since for $X^{t\geq T}(\varphi)$ we compute the predictions at different times during the stay in a context and average the results over all predictions the predictions can have lower results, but don't necessarily have to as seen in other diagrams.

Recall $X(\varphi)$ behaves very much like $X^{t\geq T}(\varphi)$ when looking at recall, with $X(\varphi)$ delivering better results for probability thresholds $p \leq 0.4$ and $X^{t\geq T}(\varphi)$ delivering better results for probability thresholds $p \geq 0.6$. The difference is about 0.05 both times. Recall is very low for $X^{T\leq t\leq T+\Delta T}(\varphi)$ again hinting at a high rate of FNs. Since we will count a FN whenever a context switch occurs that we predicted not to happen we can easily derive the reason for the bad performance. Since if the probability being reached in a time window is not enough to produce a positive prediction if the time window size is too

small we will almost always predict a negative result. For all outgoing transitions but one this of course will be true in the worst case of a wrong prediction. This also explains the good performance when looking at specificity:

Specificity $X(\varphi)$ again behaves very much like $X^{t \geq T}(\varphi)$ with $X^{t \geq T}(\varphi)$ delivering better results for probability thresholds $p \leq 0.3$ and $X(\varphi)$ delivering better results for probability thresholds $p \geq 0.4$, the difference is as much as 0.05. For probability thresholds between $0.1 \leq p \leq 0.5$ $X^{T \leq t \leq T+\Delta T}(\varphi)$ performs best, for the rest about the same as the other operators. This again suggests a high rate of TNs in comparison to the number of FPs. The reasons for this are already outlined in the previous paragraph.

Conclusion In the Dartmouth data set there seems to be no real benefit of taking transition time into account unless focusing on accuracy. For all other metrics $X(\varphi)$ performs about the same as $X^{t \geq T}(\varphi)$ with $X^{T \leq t \leq T+\Delta T}(\varphi)$ performing worst. The size of the time window has a huge impact on the performance of this operator. But since with an unlimited time window this operator behaves exactly like $X^{t \geq T}(\varphi)$ we know that this operator can perform quite well if the time window is chosen right. Choosing a time window of several days though did not seem to be a realistic time window. In general the Dartmouth data is not a very good data set for taking time into account of durations of stay in contexts since future contexts do not really seem to depend a lot on the duration of stay in a context but more on the transition probabilities to future contexts.

We will now take a look at the most likely next operators for both data sets.

Accuracy When examining the accuracy evaluation in figure 7.31 we notice that $\hat{X}$ performs about as well as $\hat{X}^{t \geq T}$. Both operators accuracy values are in the interval $[0.8, 0.9]$. Before probability threshold $p = 0.4$ $\hat{X}^{t \geq T}$ performs slightly better, after $p = 0.4$ $\hat{X}^{t \geq T}$ performs slightly better. The difference is very small though, only 0.02 on average over all probability thresholds. $\hat{X}^{T \leq t \leq T+\Delta T}$ performs worse in comparison to the other operators with an average accuracy of about 0.7 for all probability thresholds. This means that in comparison to the number of TPs and TNs there are many FPs and FNs. With increasing probability threshold p the amount of FPs and FNs still gets larger. The reason behind this is the time window being used for $\hat{X}^{T \leq t \leq T+\Delta T}$. The argument is the

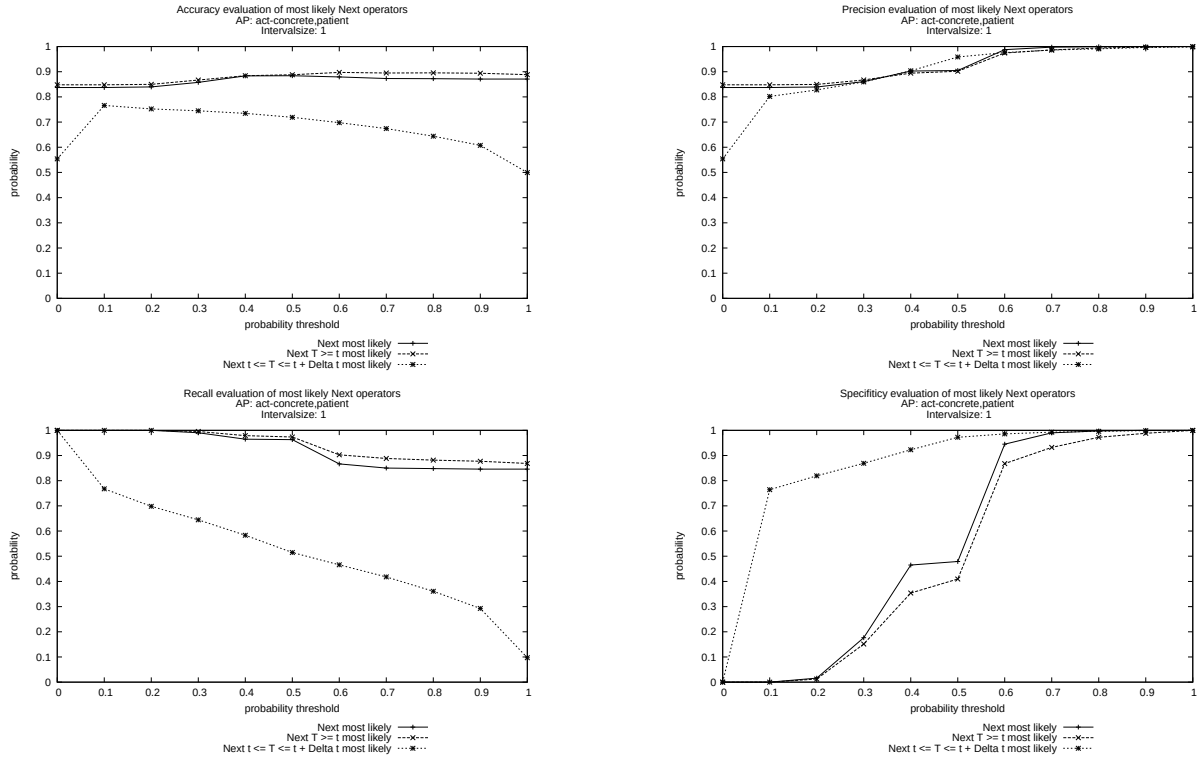

Figure 7.31: Operator prediction quality comparison for most likely Next operators in the Mainkofen data set

same as in the previous section.

Precision Precision is between 0.8 and 1.0 for all operators. So we can conclude that the number of TPs is very high and we have only very few FPs for all most likely Next operators with the Mainkofen data set.

Recall Recall is very good for $\hat{X}$ and $\hat{X}^{t \geq T}$ with values between 1.0 and 0.85. $\hat{X}^{t \geq T}$ performs better with a difference in recall of about 0.05. This means that the number of FNs is very low in comparison to the number of TPs for these two operators. $\hat{X}^{T \leq t \leq T + \Delta T}$ performs significantly worse which is again based on the same facts we established earlier. Because of the time window size a lot of FNs are created, leading to a degradation in recall quality.

Specificity The exactly opposite is true again for specificity - $\hat{X}^{T \leq t \leq T+\Delta T}$ clearly performs best with a specificity value of higher than 0.8 starting from probability threshold $p \geq 0.2$. This is also based on the size of the time window for this operator. $\hat{X}$ performs best in this evaluation directly followed by $\hat{X}^{T \leq t \leq T+\Delta T}$. There is a huge leap for specificity values for these two operators starting from $p \geq 0.5$ The reason behind this is the property of the most likely operators. Most transitions with lower transition probabilities will correctly be predicted not to be the succeeding contexts. In contrast to this given a high transition probability actually the probability usuallly is very high for a transition to be taken. Starting from $p \geq 0.6$ we can witness this behaviour.

Conclusion Unfortunately these results also tell us that the Mainkofen data set is highly dependant on transition probabilites and not so much on the time already spent in a state for all transitions with transition probabilities of higher than 0.6. The only downside of the most likely next operators comes into play when looking at specificity for probability thresholds of $p \leq 0.5$.

In figure 7.32 we display the evaluation results for the most likely next operators using the Dartmouth data set.

Accuracy Taking a look at the accuracy evaluation in figure 7.32 we examine the most likely next operators in the Dartmouth data set. In total $\hat{X}^{T \leq t \leq T+\Delta T}$ performs best especially for probability thresholds $p \geq 0.6$. Again here we can witness the almost perfectly linear steepness of this operator which is caused by the interval size granularity. For $p \geq 0.1$ we get an accuracy of 0.75 to 0.85 for $\hat{X}^{T \leq t \leq T+\Delta T}$. $\hat{X}$ performs about the same for $0.1 \leq p \leq 0.6$ but significantly worse from $p \geq 0.6$. The difference is up to 0.25. $\hat{X}^{t \geq T}$ performs slightly worse than $\hat{X}$ with a maximum of 0.05 lower in value.

Precision The evaluation results for precision show that we get the highest precision values when using $\hat{X}$ with precision values of over 0.8 for probability thresholds $p \geq 0.5$. With increasing probability threshold p the difference between $\hat{X}$ and the other two operators gets bigger and bigger, up to a difference in precision of 0.7. $\hat{X}^{t \geq T}$ performs worse than $\hat{X}$ with an average precision of about 0.8 over all probability thresholds. $\hat{X}^{T \leq t \leq T+\Delta T}$ clearly performs worst again with a precision of about 0.2 on average over all

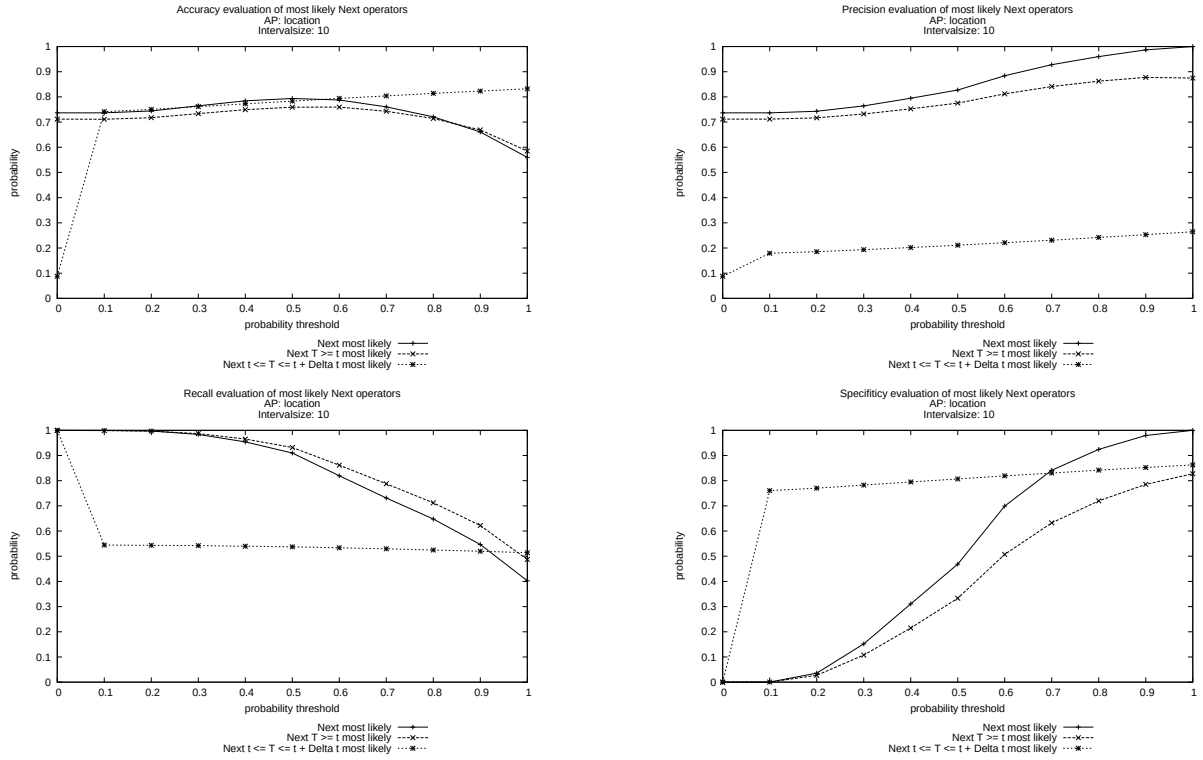

Figure 7.32: Operator prediction quality comparison for most likely Next operators in the Dartmouth data set

probability thresholds. The low precision values speak for a high rate of FPs in contrast to TPs for $\hat{X}^{T \leq t \leq T+\Delta T}$. Since the transition times are spread over a wide area of time this result is not so surprising anymore - when only looking at a relatively small (in this case 2 hours) time window only a fraction of less than 70% of all context switch times can be observed. That is why the context predictor does not take later context switches on a discrete probability density function into account, leading to many FPs in contrast to the number of TPs. The reasons for the almost linear behaviour of the curve have already been discussed earlier.

Recall Examining the results for recall for the Dartmouth data set we again see that $\hat{X}^{T \leq t \leq T+\Delta T}$ delivers the worst performance with values of about 0.55 over all probability thresholds. This means we get a high ratio of FNs compared with TPs. This again is caused by the size of the time window. The reasons for the almost linear behaviour of the curve have already been discussed earlier. This time $\hat{X}^{T \leq t \leq T+\Delta T}$ performs best with

a difference of up to 0.1 to $\hat{X}$. So there is definitely a difference in recall when using the most likely operators with and without taking time into consideration.

Specificity $\hat{X}$ clearly performs better than $\hat{X}^{t \geq T}$ for all probability thresholds p. The difference is as much as 0.15 at $p = 1.0$. $\hat{X}^{T \leq t \leq T+\Delta T}$ performs best of all operators for $0.0 \leq p \leq 0.7$ and again displays the linear behaviour already mentioned with an average of about 0.8 for all probability thresholds. This means that the number of TNs is very high in comparison to the number of FPs. The reasons behind this have again already been explained: there are hardly any FPs generated in comparison with TNs since the time window is a very precise instrument for sorting out future contexts. The number of TNs for this operator is very high.

Conclusion Again there is a tradeoff between different metrics here. Depending on which metric is important one can choose between the different operators as each operator seems to maximize one metric. No metric is best overall here. This again speaks against only taking a look at the transition probabilities.

Chapter 8

Conclusions

We successfully connected the fields of Model Checking and context prediction by showing how to transform transition systems generated from user traces to Semi-Markov chains and at the same time allowing dynamic system models for Model Checking. We successfully applied Model Checking on Semi-Markov chains generated by user traces by specifying two time based operators $X_{P\cong p}^{T\geq t}(\varphi)$ and $X_{P\cong p}^{t\leq T\leq t}(\varphi)$ and one static operator $X_{P\cong p}(\varphi)$. The evaluation results show good to very good results.

We also introduced new operators returning the most probable next contexts; for static use $\hat{X}$, with a dwell time parameter $\hat{X}^{t\geq T}$ and with time window parameters $\hat{X}^{T\leq t\leq T+\Delta T}$. The evaluation results show that these operators perform very well for both data sets. Additionally we developed a new algorithm for the computation of $\Phi U_{P\cong p}^{t\leq T\leq t+\Delta t}\Psi$.

In section 7.5 we could show that the overhead of taking the time spent in the current context into account is worth taking it into account: The prediction results are in general better than taking only a look at the transition probabilities. This is already achieved even though it is highly questionable if the Dartmouth data set is a suitable data set for taking context switch times into account. The benefit of taking time into account with the two data sets is not really high though, we could show an improvement in prediction quality of about 5-10%.

In order to really show the benefits of our approach in the future we propose to find real world data sets that especially require the taking into account of the time already

spent in a context; the best data sets would be data sets that depend less on transition probabilities and far more on the factor time.

Another result of our research is that using too many atomic propositions in combination with each other can yield worse results than choosing a lesser amount. This is because by the increased search space provided by the additional amount of data there hardly are any previous user traces going along the same paths. This leads to many more transitions in the transiton system, but with lower transition weights for each transition, creating uncertainty. At the same time we witnessed that using too few atomic propositions also resulted in bad evaluation results.

A very interesting fact especially holds true for the evaluation of the Until operator - using only the location property when predicting future locations yields very bad results in comparison to also using information about the current patient or the current activity. This is a clear sign that incorporating more than only location is highly beneficial for location prediction.

Chapter 9

Future work and open issues

9.1 Investigate the effects of using k-order Markov Chains

Using k-order Markov Chains for $k > 2$ could lead to a higher performance since more past states than only the last state will be part of the prediction process. Since our results are already very high we do not expect the difference here to be too high though. The tradeoff for using a higher value for k is the amount of memory needed which is exponentially rising with k.

9.2 Prediction of hierarchical context

Very interesting as well is formulating an ontology of different contexts, thereby creating a hierarchy of contexts. The results in the prediction of future contexts when looking at different levels of these hierarchies could be very interesting. For example, when looking at the contexts location={Brooklyn, Harlem, Manhattan} we could establish an ontology that describes that New York City is a more general description for the locations Brooklyn, Harlem and Manhattan. Having this ontology represented in the transition system would now allow queries for location=Brooklyn and at the same time location=New York City.

9.3 Symbolic Model Checking using PLTL

Another interesting approach would be to investigate if performing Symbolic Model Checking along with the use of e.g. probabilistic LTL yields better results in prediction. Symbolic Model Checking works by expressing the desired properties of a system by using LTL operators and actually checking if the model satisfies these properties. One technique to achieve this is to obtain a Bchi automaton that is equivalent to the model and one that is equivalent to the negation of the properties. If the model satisfies the properties the intersection of the two non-deterministic Bchi automata will be empty.

9.4 Ageing and uncertainty evaluation

Unfortunately the time frame did not allow for further evaluation of the theory we proposed for dealing with uncertainty and ageing. In the future the evaluation of these theoretical foundations could also lead to huge improvements.

Chapter 10

Appendix

Bibliography

[1] "Wikipedia, the free encyclopedia [online]." Available: http://www.wikipedia.com, October 2010.

[2] M. Kim and D. Kotz, "Modeling users' mobility among wifi access points," in *WiTMeMo '05: Papers presented at the 2005 workshop on Wireless traffic measurements and modeling*, (Berkeley, CA, USA), pp. 19–24, USENIX Association, 2005.

[3] S. K. Das, D. J. Cook, A. Bhattacharya, E. O. Heierman, and T. yun Lin, "The role of prediction algorithms in the mavhome smart home architecture," *IEEE Wireless Communications*, vol. 9, pp. 77–84, 2002.

[4] R. M. Mayrhofer, *An Architecture for Context Prediction*. PhD thesis, University of Linz, 2004.

[5] L. Vintan, A. Gellert, J. Petzold, and T. Ungerer, "Person movement prediction using neural networks," in *In First Workshop on Modeling and Retrieval of Context*, 2004.

[6] J. Petzold, *Zustandspraedikatoren zur Kontextvorhersage in ubiquitaeren Systemen*. PhD thesis, University of Augsburg, 205.

[7] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[8] L. Song, U. Deshpande, U. C. Kozat, D. Kotz, and R. Jain, "Predictability of wlan mobility and its effects on bandwidth provisioning," in *INFOCOM*, 2006.

[9] R. Begleiter, R. El-yaniv, and G. Yona, "On prediction using variable order markov models," *Journal of Artificial Intelligence Research*, vol. 22, pp. 385–421, 2004.

[10] D. Katsaros and Y. Manolopoulos, "Prediction in wireless networks by markov chains," *Wireless Commun.*, vol. 16, no. 2, pp. 56–63, 2009.

[11] R. 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 (ECHISE2005)* (T. Prante, B. Meyers, G. Fitzpatrick, and L. D. Harvel, eds.), May 2005. part of the Third International Conference on Pervasive Computing (PERVASIVE 2005).

[12] P. Nurmi, M. Martin, and J. A. Flanagan, "Enabling proactiveness through context prediction," in *In: CAPS 2005, Workshop on Context Awareness for Proactive Systems. (2005*, 2005.

[13] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in *Logic of Programs, Workshop*, (London, UK), pp. 52–71, Springer-Verlag, 1982.

[14] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Transactions on Programming Languages and Systems*, vol. 8, pp. 244–263, 1986.

[15] E. Clarke, O. Grumberg, and D. Long, "Model checking," in *Proceedings of the NATO Advanced Study Institute on Deductive program design*, (Secaucus, NJ, USA), pp. 305–349, Springer-Verlag New York, Inc., 1996.

[16] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[17] M. Kwiatkowska, G. Norman, and D. Parker, "Quantitative analysis with the probabilistic model checker PRISM," *Electronic Notes in Theoretical Computer Science*, vol. 153, no. 2, pp. 5–31, 2005.

[18] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)* (M. Bernardo and J. Hillston, eds.), vol. 4486 of *LNCS (Tutorial Volume)*, pp. 220–270, Springer, 2007.

[19] H. L. S. Younes and R. G. Simmons, "Statistical probabilistic model checking with a focus on time-bounded properties," *Inf. Comput.*, vol. 204, no. 9, pp. 1368–1409, 2006.

[20] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen, "Model-checking algorithms for continuous-time markov chains," *IEEE Trans. Softw. Eng.*, vol. 29, no. 6, pp. 524–541, 2003.

[21] G. G. I. Lopez, H. Hermanns, and J. pieter Katoen, "Beyond memoryless distributions: Model checking semi-markov chains," in *In Proceedings of the Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification, volume 2165 of LNCS*, pp. 57–70, Springer-Verlag, 2001.

[22] "Merriam-webster's online dictionary [online]." Available: http://www.merriam-webster.com/, October 2010.

[23] B. N. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *in proceedings of the workshop on mobile computing systems and applications*, pp. 85–90, IEEE Computer Society, 1994.

[24] S.-Y. Hwang, H. Wang, J. Tang, and J. Srivastava, "A probabilistic approach to modeling and estimating the qos of web-services-based workflows," *Inf. Sci.*, vol. 177, no. 23, pp. 5484–5503, 2007.

[25] G. Bahle, A. Grnerbl, P. Lukowicz, and J. Neuburger, "Context." University of Passau, Presentation, 2010.

List of Figures

6.1 Example Semi-Markov chain M 51

6.2 Ageing example Semi-Markov chain M 56

6.3 Ageing example Semi-Markov chain M' 58

6.4 Semi-Markov chain M'', originating from applying ageing on Semi-Markov chains M and M' 59

7.1 The floor plan for the Mainkofen data, as found in [25] 91

7.2 Wireless access points on the Dartmouth college campus, as found in [2] . . 92

7.3 State/property count in the Mainkofen data set 94

7.4 Absolute distribution of outgoing edges in the Mainkofen data set 95

7.5 Example of a part of a transition system for Mainkofen data using only the atomic proposition *location* 96

7.6 Cumulative distribution of outgoing edges in the Mainkofen data set 97

7.7 Cumulative distribution of outgoing edges in the Darthmouth data set . . 98

7.8 Relative distribution of transition probabilities in the Mainkofen data set . 99

7.9 Relative distribution of transition probabilities in the Dartmouth data set . 100

7.10 Relative and cumulative distribution of transition times in the Mainkofen data set . . . 101

7.11 Relative and cumulative distribution of transition times in the Dartmouth data set . . . 102

7.12 DPDF of an example transition from the Mainkofen data set . . . 103

7.13 Metrics for $X(\varphi)$ in the Mainkofen data set . . . 107

7.14 Evaluation of $X(\varphi)$ using Dartmouth data set . . . 111

7.15 Metrics for $\hat{X}$ in the Mainkofen data set . . . 112

7.16 Evaluation of $\hat{X}$ using Dartmouth data set . . . 114

7.17 Metrics for $X^{t \geq T}(\varphi)$ in the Mainkofen data set . . . 116

7.18 Evaluation of $X^{t \geq T}(\varphi)$ using Dartmouth data set . . . 117

7.19 Metrics for $\hat{X}^{t \geq T}$ in the Mainkofen data set . . . 119

7.20 Evaluation of $\hat{X}^{t \geq T}$ using Dartmouth data set . . . 121

7.21 Metrics for $X^{T \leq t \leq T+\Delta T}(\varphi)$ in the Mainkofen data set . . . 123

7.22 Evaluation of $X^{T \leq t \leq T+\Delta T}(\varphi)$ using Dartmouth data set . . . 124

7.23 Metrics for $\hat{X}^{T \leq t \leq T+\Delta T}$ in the Mainkofen data set . . . 126

7.24 Evaluation of $\hat{X}^{T \leq t \leq T+\Delta T}$ using Dartmouth data set . . . 128

7.25 Metrics for $X^{-1}(\varphi)$ in the Mainkofen data set . . . 130

7.26 Absolute error evaluation of $X^{-1}(\varphi)$ using Dartmouth data set . . . 130

7.27 Metrics for $\Phi U^{T \leq t \leq T+\Delta T} \Psi$ in the Mainkofen data set . . . 133

7.28 Metrics for $\Phi U^{T \leq t \leq T+\Delta T} \Psi$ in the Dartmouth data set . . . 134

7.29 Operator prediction quality comparison for Next operators in the Mainkofen data set 140

7.30 Operator prediction quality comparison for Next operators in the Dartmouth data set 142

7.31 Operator prediction quality comparison for most likely Next operators in the Mainkofen data set 145

7.32 Operator prediction quality comparison for most likely Next operators in the Dartmouth data set 147

List of Algorithms

1 The algorithm for computing $\Phi U^{t \leq T \leq t} \Psi$ 75

2 The algorithm for finding t_{fire} 78

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Klaus Peter Gruber)