PreCon – Expressive Context Prediction using Stochastic Model Checking*

Stefan Föll, Klaus Herrmann, and Kurt Rothermel

Institute of Parallel and Distributed Systems
Universität Stuttgart
Universitätstrasse 38, 70569 Stuttgart, Germany
{stefan.foell, klaus.herrmann, kurt.rothermel}@ipvs.uni-stuttgart.de

Abstract. Ubiquitous systems need to determine the context of humans to deliver the right services at the right time. As the needs of humans are often coupled to their future context, the ability to predict relevant changes in a user's context is a key factor for providing intelligence and proactivity. Current context prediction systems only allow applications to query for the next user context (e.g. the user's next location). This severely limits the benefit of context prediction since these approaches cannot answer more expressive *time-dependent queries* (e.g. will the user enter location X within the next 10 minutes?). Neither can they handle predictions of multi-dimensional context (e.g. activity *and* location). We propose PreCon, a new approach to predicting multi-dimensional context. PreCon improves query expressiveness, providing clear formal semantics by applying *stochastic model checking* methods. PreCon is composed of three major parts: a *stochastic model* to represent context changes, an *expressive temporal-logic query language*, and *stochastic algorithms for predicting context*. In our evaluations, we apply PreCon to real context traces from the domain of healthcare and analyse the performance using well-known metrics from information retrieval. We show that PreCon reaches an F-score (combined precision and recall) of about 0.9 which indicates a very good performance.

1 Introduction

Ubiquitous systems are expected to provide a new level of computational intelligence, where personalized services are tailored to fit the dynamic needs of humans. One big challenge in this respect is to proactively identify the individual needs of humans in the real world without requiring explicit input. As the user's needs are often connected to the physical context of human behaviour (e.g., where humans are going and what they are doing) ubiquitous systems have to be *context-aware*. However, in order to be proactive, knowledge about the user's *future* context is of major importance. For instance in mobile advertising [1], personalized information such as special offers or cultural events are delivered

* This research has been supported by FP7 EU-FET project ALLOW (contract number 213339).

Published in Ching-Hsien Hsu et al. (Eds.): UIC 2011, LNCS 6905, pp. 76-87, 2011.

to mobile users. Incorporating the future context, the relevance of the advertisements can be enhanced, e.g. based on the user's next locations on his shopping tour. Similarly, in domains like home automation or health care, awareness of future context yields more effective services [2]. However, future context is an implicit piece of information as it resides in the routines and habits inherent to our daily lives. Therefore, context prediction approaches have been devised to uncover the patterns of human behaviour and provide the gained knowledge to ubiquitous applications. The level of how intelligently applications can understand and respond to user needs is strongly connected to the expressiveness of queries a context prediction system can answer.

Current approaches to context prediction [3], [4], [5], [6], [7], [8] only allow for very simple queries for the most probable next user context (e.g. the user's next location). Hence, their expressiveness is severely limited since applications are not able to extract any information on the expected time of such a context change. We argue that intelligent ubiquitous applications need therefore the ability to submit *time-dependent queries* (e.g. "Will user A be at location x within the next 10 minutes?"). Likewise, queries for multi-dimensional context (e.g. "Will the user be executing *activity* Y at *location* X within the next 10 minutes?") must be possible. A system that can answer such expressive queries accurately, provides applications with more flexibility and allows them to act in a more goal-oriented way for their user.

We propose PreCon, a new approach to context prediction that allows time-dependent multi-dimensional context prediction queries. PreCon applies well-known methods of *stochastic model checking* [9] (used e.g. for the verification of distributed communication protocols) to the analysis and prediction of human behaviour. While classical model checking relies on fixed hand-crafted models of computer systems, our models (called *stochastic user models* (SUM) in the following) are representations of human behaviour that are learned from traces of past context changes. We represent SUMs as *Semi-Markov Chains* such that the changes in context are regarded as a stochastic process. We use *temporal logics* as a query language, enabling applications to specify expressive temporal properties on future context. For a prediction, our system verifies with which probability these properties hold on a given SUMs. Our evaluations show that PreCon delivers a high *precision* and *recall* for a real-world health-care scenario.

The rest of the paper is organized as follows. First, we give an overview of our approach in Section 2. In Section 3, we present the concepts of SUMs and describe our context prediction query language in Section 4. We then discuss the stochastic algorithms for query evaluation in Section 5. Section 6 presents the related work before we present the evaluation of our approach in Section 7 and conclude the paper in Section 8.

2 Overview of the PreCon Approach

Figure 1 gives a high-level overview of our approach. We assume that a context recognition system monitors the context of the user and records *context*

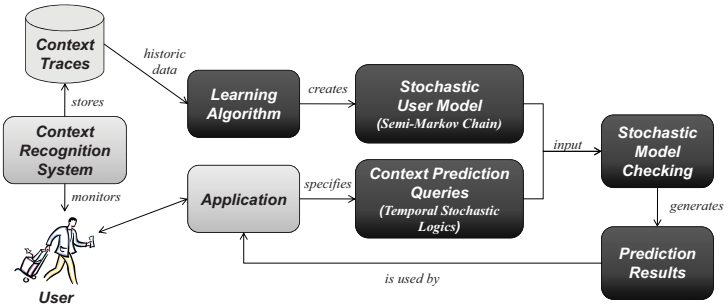


Fig. 1. Overview of the PreCon approach – Concepts specified by PerCon are shown in dark boxes

traces (time-stamped series of consecutive context changes) in user histories. For instance, a context trace may contain information about which activities have been executed at what time and location. The context traces are given as input to our *learning algorithm*, which processes them to obtain an SUM. Our framework allows context traces to be processed either in a *batch-like* fashion or in *real-time*. For the batch-like approach, the learning algorithm processes one or more context trace and creates a new SUM, while for the real-time approach, the algorithm updates the existing SUM each time a relevant context change has been observed. In both cases, the information about sequential changes in user context is used to build a *Semi-Markov Chain* (SMC) – a well-known stochastic process model that we use to represent SUMs. A SMC is a probabilistic state transition system that maintains the discrete states of the user behaviour and the associated state transition probabilities. Furthermore, the temporal characteristics of context changes (the so-called *dwell times*) are modelled by the SMC. This is the key to PreCon's concept of time-dependent queries.

Applications can specify *context prediction queries* using different temporal operators that are part of a temporal stochastic logic. This query language provides well-defined semantics to express *reachability properties* (e.g. will the user arrive at a certain location) and *invariant properties* (e.g. will the user stay at a certain location). A context prediction query is evaluated on an SMC to calculate the probability with which the specified properties hold. A querying application can specify a probability threshold with which the resulting probability is compared, and a *true* or *false* is returned depending on the outcome. Finally, the querying application may use this result to take proactive decisions in terms of, e.g. user interaction and context-aware services, to enhance the user's experience. In the following sections, we will investigate each element of PreCon in turn.

3 Stochastic User Model

People follow varying behavioural patterns in their daily lives, such that real world user behaviour can not be described in a deterministic way. Consequently, we require a user model that is able to deal with this probabilistic nature of human behaviour. In the following, we give a precise formal definition of the SUM. Subsequently, we present our approach for learning such a SUM from observations recorded in real-world context traces.

3.1 Semi-Markov Chain

We represent an SUM as a Semi-Markov Chain (SMC) [10]. In general, Markov Chains are a popular means for describing stochastic processes with discrete state spaces. In addition to that, SMCs specify a so-called *state dwell time* – an arbitrary probability distribution that is associated with every state transition specifying the amount of time spent in a given state. Formally, a SMC M is a 3-tuple defined as:

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

where S is the state space, $p : S \times S \rightarrow [0, 1]$ with $\forall s \in S : \sum_{s' \in S} p\,(s, s') = 1$ is the transition probability function, and $h : (s, s', t) \mapsto [0, 1]$ with $t \in \mathbb{R}^+$ represents the distribution of dwell times associated with a state transition $(s, s') \in S \times S$. For $h : (s, s', t)$, we will also write $h_{s,s'}\,(t)$ for brevity reasons. The SMC allows us to describe a user's behaviour in the following manner: At each point in time, a user is in a state $s \in S$ that is identified by his current context (cf. Section 3.2). While the user acts in the real world, his context changes and his SMC moves to a new state $s' \in S$ representing the new context. s' is called the *successor state* of s, and s' is visited with a certain probability $p(s, s')$. Before leaving the current state s, s is active for a limited amount of time (the dwell time represented by $h_{s,s'}\,(t)$). During this time period the user's context does not change.

3.2 Learning a SMC

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

User States A user state $s = (c_1, ..., c_n)$ is an n-dimensional vector of context information. Each component c_i of s is of a specific *context type* C_i, and $C_1, ..., C_n$ are the context types known to the system with domains $Dom(C_1), ..., Dom(C_n)$.

E.g. c_i may be an integer value from $Dom(C_i) = \mathbb{N}^+$, and C_i may be the *ambient temperature* type. Other types could be *location* and *activity*, and the corresponding domains could be enumerations of possible activities and symbolic location identifiers respectively. For example, the state $s = (\text{"}\textit{meeting room}\text{"}, \text{"}\textit{give presentation}\text{"}) \in Dom(Location) \times Dom(Activity)$ describes the fact that the user executes the activity *give presentation* in a location referred to as *meeting room*. Whenever a combination of context information $(c_1, ..., c_n)$ is detected that has not already been encountered for the specific user, a new user state $s = (c_1, ..., c_n)$ is added to the SMC.

Transition Probabilities A concrete series of consecutive user states is represented as a stochastic process of random variables $X_1, X_2, X_3, ...$, where X_i refers to the state occupied after the i-th state transition. In order to learn the state transition probabilities, we assume the Markov property: The probability $p(s, s')$ for the state s' to be visited next only depends on the current state s, and is independent of all previous state changes. This assumption can be extended such that $p(s, s')$ depends on the k last visited states (k-order Markov models [3]) if needed, and PreCon operates on these more general k-order models. However, for simplicity, we assume $k = 1$ here. The math is essentially the same.

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

Dwell Time Distribution The dwell time in state s is modelled as a random variable D_s. We learn the probability distribution of D_s conditioned on each transition such that $h_{s,s'}(t) = P(D_s = t | X_{n+1} = s', X_n = s)$. For this purpose, we observe the time periods that pass between consecutive changes in user state. In order to limit the storage and computation overhead, we apply a discretization and divide time into intervals of equal size Δt, such that the i-th time interval is defined as $I_i = [i \cdot \Delta t, (i+1) \cdot \Delta t)$. The distribution $h_{s,s'}$ can then be derived as follows: Let $w^i_{s,s'}$ be the number of transitions (s, s') that occurred in the interval I_i such that $w_{s,s'} = \sum_i w^i_{s,s'}$ is the total number of observed transitions (s, s'). Then the probability for spending exactly time t in state s before leaving to successor state s' is calculated as

$$h_{s,s'} : t \mapsto \frac{w_{s,s'}^{\lfloor \frac{t}{\Delta t} \rfloor}}{\Delta t \cdot w_{s,s'}}. \tag{1}$$

In equation 1, we use Δt as a normalization factor to ensure that $\int_0^\infty h_{s,s'}(t)\, dt = 1$ for the cumulative distribution. As we deal with a discrete representation of

the dwell time distribution, the cumulative distribution function $\int_a^b h_{s,s'}(t)\,dt$ is computed as a sum of intervals over the probability mass function. More precisely, the cumulative probability is determined by the sum over the intervals which are enclosed by the integral ranges between a and b. As a and b may fall into discretization intervals, we interpolate the probability associated with the fraction of the corresponding intervals based on a linear function. The cumulative distribution is later used in Section 5 to determine the probability resulting from time-bounded queries.

4 Prediction Query Language

PreCon's prediction query language is based on *Continuous Stochastic Logic* (CSL) [11], a probabilistic derivative of branching-time temporal logics (applied in classical model checking). CSL provides operators for verifying *temporal properties* of probabilistic state transition systems. Applications construct queries from these *temporal operators* and submit them to PreCon. The operators are then evaluated on the learnt SMC to verify the specified properties.

The state space S can be traversed by going from one state to the next as the transitions among the states permit. The resulting series of visited states (called a *path*) models one possible temporal behaviour of the user. For a context prediction, PreCom starts at the state $s \in S$ the user currently occupies in the real world and evaluates the given query from there, possibly considering all possible paths starting at s (depending on the temporal operators in the query). The query language is defined as follows:

Let $p \in [0,1]$ be a probability threshold, let $\triangleleft \in \{\leq, \geq\}$ be a comparison operator, let $t \in \mathbb{R}^+$ be a time bound, and let $(C_i, c \in Dom(C_i))$ be a contextual value c of type C_i. Queries can be composed from CSL using the following grammar:

A query is a temporal-logic formula Φ with

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

where φ is a *path formula* defined as

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

Using CSL, we can investigate *reachability properties* (using operators X and F) and *invariant properties* (using operators G and U) of future user behaviour. X is the *Next* operator. It evaluates a condition Φ on all immediate successor states of the current user state s. Φ is expressed as a name-value pair (C_i, c) consisting of the name of a context type C_i (e.g. *location*) and a specific context value c (e.g. *office*). The query "Will the next location be the office?" can be expressed by applying the *Next* operator to $\Phi = (location, office)$, resulting in $X(location, office)$. F is the *Eventually* operator and can be used to verify if a condition Φ holds in any state reachable from s through paths in the SMC. G is the *Globally* operator and can be used to check if the condition Φ holds in

every state on all paths starting in s. U is the *Until* operator and expresses that eventually Φ_2 must hold and Φ_1 must hold on all paths starting at the current state until Φ_2 holds.

Time is a first order construct of the prediction query language. All operators are associated with a time constraint t, defining an upper bound on the time, which may pass until the desired property holds. Having such a time bound and using the dwell time distributions to evaluate time-bounded queries enables us to formulate time-dependent queries.

The *raw* predictions are always probabilistic in nature when a query is evaluated. So the answer of the model checking algorithm is of the form "The user enters his office within the next 10 minutes with probability 0.74". A querying application, however, usually expects a *true* or *false* as an answer. Therefore, the calculated probabilities are compared to a probability threshold p, which is expressed in the subscript of a query formula ($\mathbb{P}_{\triangleleft p}(\varphi)$). The querying application specifies this probability and gets a boolean result depending on whether the outcome of the query evaluation exceeds the threshold or not[1].

In Table 1, we give some examples for behavioural properties which can be expressed as CSL formulas. The examples demonstrate the range of different use cases for context predictions, including queries with different semantics and context types.

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

Table 1. Examples of context prediction queries

The probability threshold p is an application-dependent value to influence the trade-off between *false positives* (queries that evaluate to *true* but prove to be false later on) and *false negatives* (queries that evaluate to false but actually become true in reality): A higher threshold reduces the number of false positives

[1] Applications can also access the raw prediction result in case they require more complex threshold comparisons.

and increases the number of false negatives. A lower threshold has the opposite effect. Consequently, the concrete threshold defines the ratio of false negative and false positives that the application is willing to accept. The choice for the threshold is dependent on the application semantics. For example, in security critical applications it is usually beneficial to prepare for exceptional cases even if they might not occur. Hence, such applications may tolerate a higher number of false positives rather than false negatives. On the contrary, a large number of false positives may negatively impact the satisfaction of a user in an application that delivers advertisements based on his predicted future location. In this case a higher probability threshold is beneficial to prevent the user from being overwhelmed by irrelevant advertisements.

5 Query Processing

Classical model checking algorithms assume static state transition systems, where the system is analysed at design-time and behavioural properties are only studied at state entry times. In our case, the system that is subject to the verification is dynamic. In particular, the probability resulting from the evaluation of a query is depending on the time Δd, that has passed since the current state s was entered. Therefore, we have to extend the standard model checking approach to account for Δd (referred to as the *running dwell time* in the following) by devising new ways of evaluating the temporal operators X and U. This is sufficient since it can be shown that, F and G can be expressed using the X and U operators [9]. For example, the reachability property $F^{\leq t}\Phi$ can be transformed to the equivalent expression $(true\, U^{\leq t}\Phi)$. We refer to X and U as the *basic operators* in the following. Arbitrarily complex temporal-logic formula can be evaluated in a bottom-up manner based on a tree representation [9] using only the basic operators. Thus, evaluating a query requires two things:

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

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

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

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

In the following, we will present the evaluation approach for the two basic operators in detail. Let i be the index of the last state transition that was

observed, such that $X_i = s$ denotes the current state and $D_s = \Delta d$ is the current dwell time that has passed in this state. As a common basis for the computations, we define the probability for moving from the state s to a successor state s' within time t using the information present in the SMC as follows:

$$P\left(X_{i+1} = s', D_s \leq \Delta d + t | X_i = s, D_s > \Delta d\right) \tag{2}$$

$$= \frac{P\left(X_{i+1} = s', \Delta d < D_s \leq \Delta d + t | X_i = s\right)}{\sum_{s' \in S} P\left(X_{i+1} = s', D_s > \Delta d | X_i = s\right)} \tag{3}$$

$$= \frac{p(s,s') \cdot \int_{\Delta d}^{\Delta d + t} h_{s,s'}(x)\, dx}{\sum_{s' \in S} p(s,s') \cdot \int_{\Delta d}^{\infty} h_{s,s'}(x)\, dx} \tag{4}$$

We use Baye's rule to transform formula (2) into (3), which is free of the dwell time distribution in the conditional probability. Thus it can be computed using the state transition probabilities and the dwell time distribution present in the SMC (Equation 4).

5.1 Next Operator *X*

The next operator limits the search space for the satisfaction of property φ to the immediate successor states of the current state s. Due to the running dwell time, we have to consider the dwell time distribution only from the time Δd onwards and express this using a subscript in $X^{\leq t}_{\Delta d}$. We extend the model checking approach given by Lopez et al. [12] accordingly, as follows:

$$P(X^{\leq t}_{\Delta d}(\varphi)) \tag{5}$$

$$= \sum_{s' \in S \wedge s' \models \varphi} P\left(X_{i+1} = s', D_s \leq \Delta d + t | X_i = s, D_s > \Delta d\right) \tag{6}$$

$$= \frac{\sum_{s' \in S \wedge s' \models \varphi} p(s,s') \cdot \int_{\Delta d}^{\Delta d + t} h_{s,s'}(x)\, dx}{\sum_{s' \in S} p(s,s') \cdot \int_{\Delta d}^{\infty} h_{s,s'}(x)\, dx} \tag{7}$$

We can calculate $P(X^{\leq t}_{\Delta d}(\varphi))$ directly, using Equation 4. The denominator is the probability of reaching arbitrary next states in greater than Δd time units, whereas the nominator reduces this probability to states where the φ holds. This ratio represents the desired probability considering the running dwell time Δd.

5.2 Until Operator *U*

For the until operator, the satisfaction of the property φ must be evaluated along all paths which can be reached from the current state within the given time

bound. The exploration of the state space is therefore not necessarily bound by the immediate successor states. Again, we extend the approach of Lopez et al. [12] by additionally considering the running dwell time Δd. This is expressed using a subscript in $U^{\leq t}_{\Delta d}$. The probability for the satisfaction of $U^{\leq t}_{\Delta d}$ can then be calculated as follows:

$$P(\Phi_1 U^{\leq t}_{\Delta d} \Phi_2) = F_a(s, s', t, \Delta d) \tag{8}$$

$$F_a(s, s', t, \Delta d) = \begin{cases} 1, \text{ if } s \models \Phi_2 \\ \frac{1}{\sum_{s' \in S} p(s,s') \cdot \int_{\Delta d}^{\infty} h_{s,s'}(t) dt} \cdot \sum_{s' \in S} \int_{\Delta d}^{\Delta d + t} p(s, s') \\ \qquad \cdot h_{s,s'}(x) \cdot F_b(s, s', t - x) dx, \text{ if } s \models \Phi_1 \wedge \neg \Phi_2 \\ 0, \text{ otherwise} \end{cases} \tag{9}$$

$$F_b(s, s', t) = \begin{cases} 1, \text{ if } s \models \Phi_2 \\ \sum_{s' \in S} \int_0^t p(s, s') \cdot h_{s,s'}(x) \cdot F_b(s, s', t - x) dx \\ \qquad \text{, if } s \models \Phi_1 \wedge \neg \Phi_2 \\ 0, \text{ otherwise} \end{cases} \tag{10}$$

The extension of the standard algorithm results in two functions F_a (9) and F_b (10). Function F_a is used in the first step of the verification, starting from the current user state s taking the running dwell time in s into account. Function F_a uses F_b for calculating the probability over all the possible paths starting at s. F_b calculates the probabilities recursively using convolution of dwell time distributions, taking into account that a state may be left at each point in time within the remaining time horizon.

6 Related Work

Context prediction in pervasive computing has attracted much attention in recent years. Maryhofer proposed a general architecture for context prediction [2] and used methods from time series analysis (e.g. ARMA) for prediction. Other approaches have used Markov Models [5], [3], [8], compression algorithms [3], [7], n-gram tries [6], or machine learning techniques [4] to predict context. However, the predictions supported by these approaches only compute the most probable next context in a single-dimensional context space. They do not allow queries for temporal relations in a multi-dimensional context space. Semi-Markov Chains as we use them have also been employed by Lee and Hou for an analysis of transient and stationary features of user mobility on the Dartmouth campus network [13]. However, they also did not consider a sophisticated query language that supports temporal relations. PreCon is the first system to investigate the application of temporal logics as a powerful and expressive query language for context predictions.

For the calculation of the predictions, we adopt the techniques from the field of stochastic model checking [14], which has been thoroughly studied by

Baier et al. for Continuous Time Markov Chains [11]. Lopez et. al have worked on the extension of the model checking algorithms for Semi Markov Chains [12]. In our approach, we leverage on their results. However, we extend their original model checking algorithms by including the running dwell time of the states, allowing for accurate real-time predictions. In contrast to classical model checking approaches, PreCon does not rely on a design-time specification of a state transition system. Instead, we apply a learning algorithm for building an SMC in an online fashion. This allows us to incorporate newly available data at any time, making the predictions more accurate.

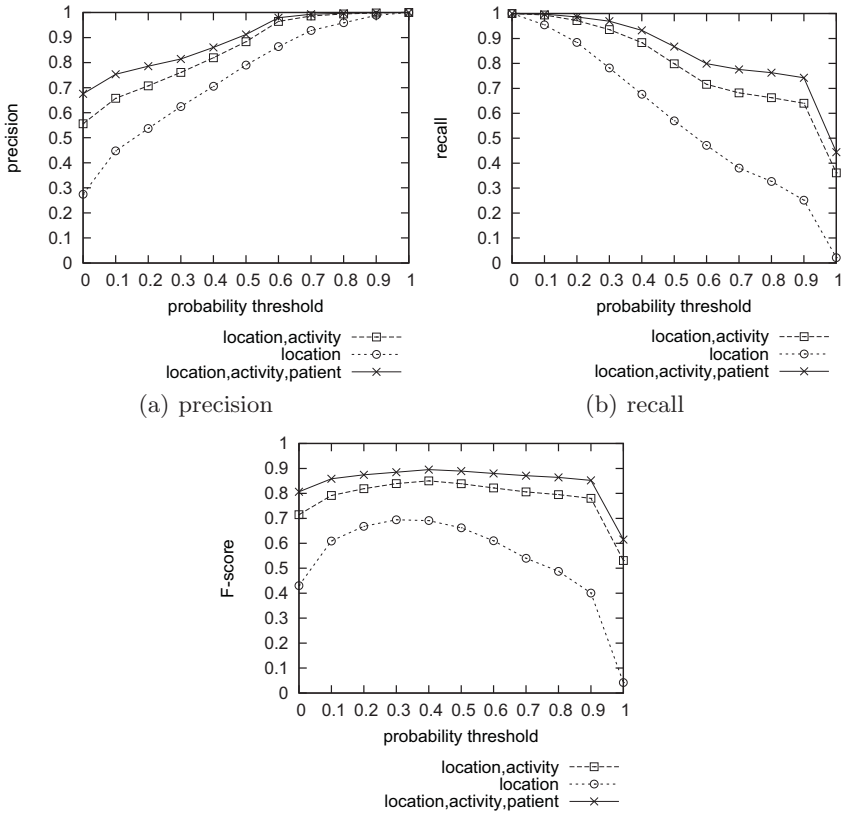
The combination of SMCs, a query language based on temporal logic, and the online learning approach represents an important new step in the area of context prediction.

7 Evaluation

We have evaluated PreCon using real-world context traces from a case study in a German geriatric nursing home. The nursing home is an intensive care station for elderly people suffering from dementia and other old-age diseases. The patients are accommodated in rooms on a nursing ward, where they receive care from nurses throughout the day. Each nurse visits patients in different rooms and performs treatment activities (e.g., the patient morning hygiene). PreCon predicts the future context of the nurses, in order to optimize the tasks scheduled by an intelligent workflow management system. The integration of context prediction into the scheduling decisions is part of the European research project ALLOW [15]. In order to obtain context traces, the nurses were accompanied over the course of 25 days during 3-5 hours in the morning shift. The traces consist of time-stamped entries of (1) the activities performed by nurses (2) the locations of their visits and (3) the ids of the patients they took care of. Thus, the records define a time series of multi-dimensional context, where each entry denotes a discrete change of context associated with a nurse. Given the context traces, PreCon learns a SMC to represent the behaviour of each nurse. In order to evaluate the impact of different context types on the prediction outcome, we varied the types of context used to learn the SMCs. We investigated three different state spaces, i.e., $s \in Dom(Location), s \in Dom(Location) \times Dom(Activity)$ and $s \in Dom(Location) \times Dom(Activity) \times Dom(Patient)$.

It is important to note that comparisons with existing context prediction systems are not possible at this time as they cannot produce the type of temporal predictions generated by PreCon. Since PreCon is the first system to venture into this area, we use metrics from the area of information retrieval (as explained in the following) to asses the general performance of PreCon.

Metrics We performed the evaluations using the metrics *precision*, *recall* and *F-score* known from the area of information retrieval. If a query result exceeds the probability threshold, we count it as either *true positive* (TP) or *false positive* (FP), depending on whether the prediction matches the real-world context.



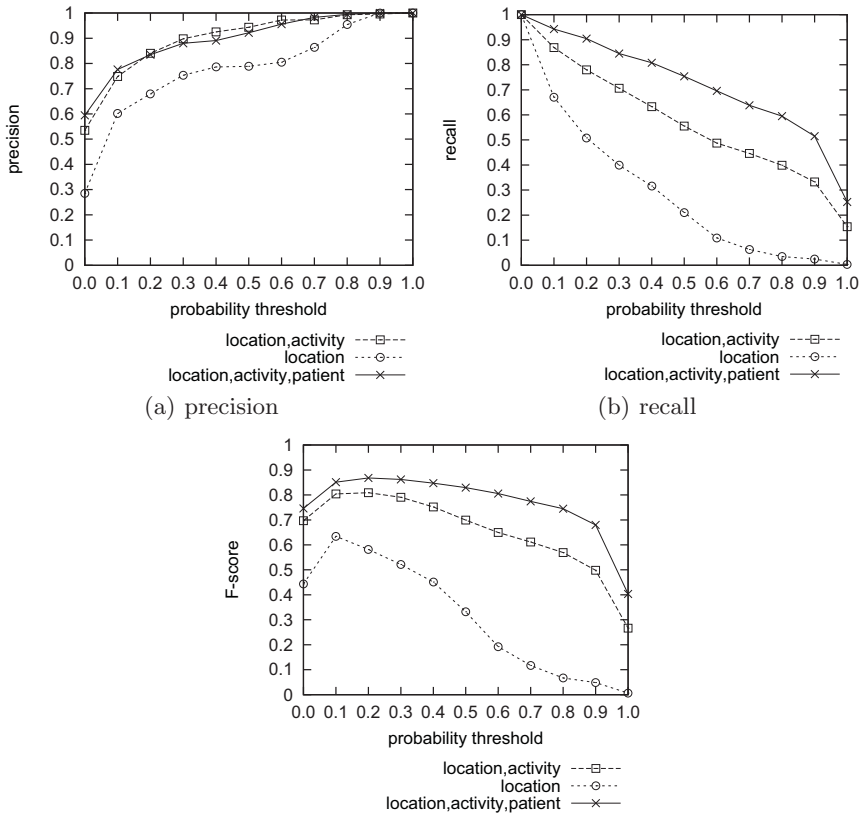
(a) precision (b) recall

(c) F-score

Fig. 2. Evaluation results for the Next operator

Otherwise, in case the prediction remains below the probability threshold, we distinguish between *true negatives* (TN) or *false negatives* (FN). We count the occurrences of (TP), (FP), (TN), and (FN) in order to calculate the metrics. This way, we can evaluate the influence of a varying probability threshold p on the *precision* defined as $\frac{TPs}{TPs+FPs}$ as well as on *recall* defined as $\frac{TPs}{TPs+FNs}$. While precision is a measure of the exactness of the predictions, recall is used to quantify the completeness of the predictions. Additionally, we evaluate the *F-score* which gives a combined measure of both and is defined as $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{recision} + \text{recall}}$. The performance of these metrics gives important insight in how proactive applications are affected by a choice of the probability threshold as will be discussed in the next subsection for different queries.

Basic Queries We evaluated our approach for two exemplary queries $\mathbb{P}_{\triangleleft p}(X_f^{\leq t}(\varphi))$ and $\mathbb{P}_{\triangleleft p}(\ F^{\leq t}\Phi_2) = \mathbb{P}_{\triangleleft p}(\ true U^{\leq t}\Phi_2)$. We generated queries for the future location of a user. The time constraint associated with these queries is set to $t = 10$



(a) precision (b) recall

(c) F-score

Fig. 3. Evaluation results for the Until operator

minutes. We evaluated the queries repeatedly, i.e, predictions were computed upon a state change and periodically after $\Delta E = 10$ seconds have passed in a state. The results discussed in the following show the average of 2000 query evaluations.

Figures 2(a)-(c) illustrate the results for the *Next* operator $X_f^{\leq t}(\varphi)$. Figure 2(a) shows the impact of a varying probability threshold on the precision metrics. As expected, the precision gains from an increase of the probability threshold. The reason is that the number of FP decreases because an increasing portion of the predictions with a low probability is discarded. At the same time, we can observe the highest precision if states are composed of the multi-dimensional context "concrete activities", "location" and "patient". In contrast, the precision remains lowest when states only contain location information (single-dimensional). Hence, the evaluation results show that additional context is relevant to discriminate user states such that more accurate predictions can be expected. Figure 2(b) shows the recall as a function of the probability threshold. The results

are reciprocal to the precision results: For an increasing probability threshold the number of FN increases, as more and more predictions are discarded that actually match the future behaviour. This result illustrates the trade-off between precision and recall for different values of the probability threshold: If the threshold is increased to guarantee more reliable predictions, the risk of discarding correct predictions with a low probability rises. Figure 2(c) shows the F-score and reveals the characteristics of this trade-off: The F-score rises as the threshold increases from 0 to 0.4. Up to this threshold, the gain from a higher precision outweighs the loss in recall. However, for a threshold higher than 0.4, the loss due to the loss in recall becomes more significant, so that the score is negatively affected. For applications that are interested in a good trade-off, we therefore recommend $p = 0.4$ as a probability threshold. In this case, we achieve a F-score of 0.86, which indicates a very good performance.

Figures 3(a)-(c) depict the evaluation results for the until operator. Figure 3(a) shows that the precision significantly gains already for lower thresholds $p > 0$. The reason is that Φ_2 can be fulfilled in all states reachable within the time constraint. Thus, the chance to encounter the expected context is increased. In contrast, the same chance is limited to the state successors in case of the next operator. At the same time, the recall is highly reduced by an increasing threshold as shown in Figure 3(b). Since the evaluated queries also address future context, which appears in states reachable over multiple transitions that are uncertain to occur, a significant amount of predictions has only a minor probability. This large number causes a lot of FNs in total, so that a significant amount of correct predictions is discarded. This observation is also reflected in Figure 3(c), which shows the F-score. For a threshold $p > 0.2$ the high loss in recall dominates the gain in precision. Compared to the next operator, the until operator is more sensitive to the choice of the probability threshold. We therefore recommend $p = 0.2$ as a threshold for applications to deal with the inherent trade-off for the until operator. In this case, a good result with an F-score of 0.87 can be achieved.

8 Conclusion

We presented PreCon, a novel approach to context prediction that enables intelligence in ubiquitous system by allowing for much more expressive queries than existing systems. In PreCon, user behaviour is represented by Semi-Markov Chains (SMC), and temporal-logics is used as a query language. We extended well-known model-checking techniques to deal with the online character of context predictions and to allow for continuous learning of SMCs. PreCon's query language provides a powerful means for applications to pose temporal queries for reachability and invariant properties of future context. Thus, PreCon goes far beyond existing approaches and represents a new class of context prediction systems that enable intelligent ubiquitous applications to take much more educated decisions.

We evaluated PreCon based on a real-world case study in the area of healthcare using metrics from information retrieval and showed that it exhibits a good performance. Moreover, our evaluations yielded indications for choosing sensible parameters for different classes of applications.

In our future work, we will extend our approach to probabilistic LTL as another variant of temporal logics. This will enable us to define further types of queries, which can not be expressed in CSL. Furthermore, we are working on a distributed context prediction system where prediction models are cached on mobile devices such that a user always has access to the future context that is relevant for him.

References

1. Krumm, J.: Ubiquitous advertising: The killer application for the 21st century. Pervasive Computing **10**(1) (2011) 66–73
2. Mayrhofer, R.M.: An Architecture for Context Prediction. PhD thesis, Johannes Kepler University of Linz (2004)
3. Song, L., Kotz, D., Jain, R., He, X.: Evaluating next-cell predictors with extensive wi-fi mobility data. IEEE Transactions on Mobile Computing **5** (2004) 1633–1649
4. Anagnostopoulos, T., Anagnostopoulos, C., Hadjiefthymiades, S., Kyriakakos, M., Kalousis, A.: Predicting the location of mobile users: a machine learning approach. In: Proc. of the 6th Intl. Conf. on Pervasive Services. (2009)
5. Katsaros, D., Manolopoulos, Y.: Prediction in wireless networks by markov chains. IEEE Wireless Communications **16** (2009) 56–63
6. Hartmann, M., Schreiber, D.: Prediction algorithms for user actions. In: In Proc. of Intl. Conf. on Adaptive Business Information Systems. (2007)
7. Gopalratnam, K., Cook, D.J.: Online sequential prediction via incremental parsing: The active lezi algorithm. IEEE Intelligent Systems **22** (2007) 52–58
8. Davison, B.D., Hirsh, H.: Predicting sequences of user actions. In: Workshop on Predicting the Future: AI Approaches to Time Series Analysis. (1998)
9. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
10. Howard, R.A.: Dynamic Probabilistic Systems: Semi-Markov and Decision Processes. John Wiley & Sons (1971)
11. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time markov chains. IEEE Transactions on Software Engineering **29** (2003) 524–541
12. Lopez, G.G.I., Hermanns, H., Katoen, J.P.: Beyond memoryless distributions: Model checking semi-markov chains. In: Process Algebra and Probabilistic Methods. Performance Modeling and Verification. (2001)
13. Lee, J.K., Hou, J.C.: Modeling steady-state and transient behaviours of user mobility: Formulation, analysis, and application. In: Proc. of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing. (2006)
14. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation **4486** (2007) 220–270
15. Herrmann, K., Rothermel, K., Kortuem, G., Dulay., N.: Adaptable pervasive flows - an emerging technology for pervasive adaptation. In: Proc. of the Workshop on Pervasive Adaptation at the 2nd Intl. Conf. on Self-Adaptive and Self-Organizing Systems. (2008)