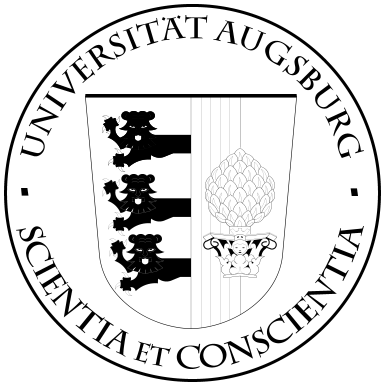
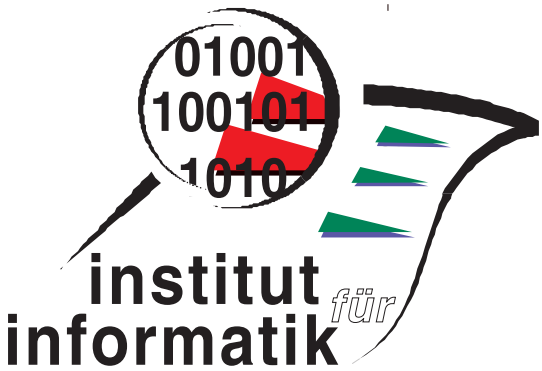
Universität Augsburg



Context Prediction Based on Branch Prediction Methods

Jan Petzold, Faruk Bagci, Wolfgang Trumler, and Theo Ungerer

Report 2003-14 Juli 2003



Institut für Informatik
D-86135 Augsburg

Context Prediction Based on Branch Prediction Methods

Jan Petzold, Faruk Bagci, Wolfgang Trumler, and Theo Ungerer

University of Augsburg
Institute of Computer Science
Eichleitnerstr. 30, 86159 Augsburg, Germany
{Petzold, Bagci, Trumler, Ungerer}@Informatik.Uni-Augsburg.DE

Abstract. Ubiquitous systems use context information to adapt appliance behavior to human needs. Even more convenience is reached if the appliance foresees the user's desires and acts proactively. This paper focuses on context prediction based on previous behavior patterns.
The proposed prediction algorithms originate in branch prediction techniques of current high-performance microprocessors which are transformed to handle context prediction. We propose and evaluate the one-level one-state, two-state, and multiple-state predictors, and the two-level two-state predictors with local and global first-level histories. Evaluation is performed by simulating the predictors with behavior patterns of people walking through a building as workload. The evaluations show that the proposed context predictors perform well but exhibit differences in training and retraining speed and in their ability to learn complex patterns.

1 Introduction

Ubiquitous systems strive for adaptation to user needs by utilizing information about the current context in which a user's appliance works. A new quality of ubiquitous systems may be reached if context awareness is enhanced by predictions of future contexts based on current and previous context information. Such a prediction enables the system to proactively initiate actions that enhance the convenience of the user or that lead to an improved overall system.

Humans are creatures of habit. Humans typically act in a certain habitual pattern, however, they sometimes interrupt their behavior pattern and they sometimes completely change the pattern. Our aim is to relieve people of actions that are done habitually without determining a person's action. The system should learn habits automatically and reverse assumptions if a habit changes. The predictor information should therefore be based on previous behavior patterns and applied to speculate on the future behavior of a person. If the speculation fails, the failing must be recognized, the speculatively initiated actions withdrawn, and the predictor updated to improve future prediction accuracy.

For our application domain we chose next location prediction instead of general context prediction. The algorithms may also be applicable for other more

general context domains, however, there exist already numerous scenarios within our application domain. Some sample scenarios may be the following:

- Smart doorplates that are able to direct visitors to the current location of an office owner based on a location-tracking system and predict if the office owner is soon coming back [11].
- Similarly, next location prediction within a smart building can be used to prepare the room which is presumably entered next by an habitant.
- Elevator prediction could anticipate at which floor an elevator will be needed next.
- Routing prediction for cellular phone systems may predict the next radio cell a cellular phone owner will enter based on his previous movement behavior.

The term "context prediction" is used by Gellersen et al. [3] when a current situational context is generated by combining different sensor information. The semantic of our use of context prediction differs from Gellersen et al.'s use. We always mean next context prediction, which is rarely investigated in ubiquitous computing environments.

To predict or anticipate a future situation learning techniques as e.g. Markov Chains [1], Bayesian Networks [6], Neural Networks [4] are obvious candidates. The challenge is to transfer these algorithms to work with context information.

The Adaptive House project [9] of the University of Colorado developed a smart house that observes the lifestyle and desires of the inhabitants and learned to anticipate and accommodate their needs. Occupants are tracked by motion detectors and a neural network approach is used to predict the next room the person will enter and the activities he will be engaged. Hidden Markov Models and Bayesian Inferences are applied by Katsiri [8] to predict people's movement. Markov Chains are used by Kaowthumrong et al. [7] for active device selection.

In our work we choose a completely different approach. Branch prediction techniques as know from high-performance processors are transferred to the domain of context prediction. Our sample application predicts the next location of people moving within an office building.

The next section shortly introduces branch prediction techniques and demonstrates the problems that arise for a transfer towards next context prediction. Section 3 describes the proposed context prediction algorithms and section 4 evaluates the predictors. The paper ends with the conclusions.

2 From Branch Prediction to Context Prediction

Branch prediction techniques as used in current high-performance microprocessors [12] stand out by their implementation simplicity, short run-time, small memory needs, and nevertheless high hit ratio of often up to 97% in the specialized application domain of predicting the outcome of a branch instruction. All these features are also highly desirable for context prediction, in particular, when context prediction is used by small battery-powered computers in ubiquitous appliances. Branch prediction algorithms are based on simple hardware operations

and table look-ups to allow a hardware implementation and prediction within a single processor cycle of a multiple Gigahertz processor. Storage needs are small because all prediction tables must be implemented on the processor chip. A high prediction accuracy is essential for the overall processor performance, because the misprediction penalty resulting from squashing of falsely fetched and executed instructions and reloading the pipeline numbers several processor cycles.

Current prediction techniques use branch execution history collected during the runtime of a program. Such dynamic branch prediction techniques started with simple one-bit predictors that use a single bit to determine the speculation direction. Speculation is always determined by the outcome of the last execution of the branch. Two-bit predictors distinguish four states stored in two bits: predict strongly taken (11), weakly taken (10), weakly not taken (01), and strongly not taken (00). If the prediction is in a strong state, then the prediction must miss two times to reverse speculation to the opposite direction. A single habitual change in the execution pattern of the branch should not reverse the prediction. The saturation counter scheme simply decrements or increments the predictor value in case of misprediction respectively prediction hit until the saturation value is reached. Two-bit predictors improve prediction accuracy over one-bit predictors in case of nested loops [12]. Multiple bit predictors need more storage space and evaluations showed that branch prediction accuracy is not improved [5]. In principle, multiple bit predictors change a learned habit more slowly and the number of used bits determines how slow the habit change occurs. Multiple bit predictors can be seen as a kind of Markov predictor [2], if the multiple bit representation is replaced by the frequencies of taken or not taken. One-, two-, and multiple bit predictors are first candidates for a transfer to context prediction.

All above-mentioned predictors use a single table that in an ideal implementation contains an entry for each branch instruction (in reality several branch instruction may be mapped to the same entry leading to undesired interferences) and consider only self-history of the specific branch. The idea of correlating predictors [10] or two-level adaptive predictors [13, 14] exploits in addition neighbor-history by a two-level approach — the first level consists in its most simple approach of a shift register that stores the outcomes of the last executed branches, and the second level is typically a two-bit predictor table. The two-level adaptive predictor schemes proposed by Yeh and Patt [14] distinguish nine different approaches to two-level adaptive predictors that use partly global (neighbor) or local (self)history in the first level and local history in the second level. These predictors are the basis of all branch prediction schemes used in current high-performance microprocessors. The two-level adaptive schemes are described in detail in [14] and [12].

The next section will show how these ideas can be transferred and applied for context prediction in ubiquitous computing environments. We exemplify the prediction techniques by movement sequences of a person that moves through several rooms of a building. The predictions in our example only concern neigh-

bor rooms, but the techniques can also be applied to predict distant target rooms. Previous movements from one room into its neighbor rooms are represented by an entry in the predictor table similar to a branch instruction. Interferences caused by address aliases, a difficult problem in branch prediction, cannot arise in most of our predictors, because each possible room change is represented by an own table entry. Branch outcomes can be one of only two states: taken or not taken. An analogy to context predictors arises if there are only two neighbor rooms. If there are more than two neighbor rooms, the analogy with branches fails, and the predictors must be enhanced.

Our predictors only predict the location contexts of a single person. If multiple persons should be observed, separate predictor tables are applied. This is no restriction as far as the movements of several persons do not interfere with each other. Such personal interferences cannot be observed by the proposed algorithms.

3 Context Prediction Algorithms

3.1 Local One-level Context Predictors

We start with the one-level context predictors that allow only to base the predictions on local movements from one room to its neighbor rooms.

1-State Context Predictor. The predictor stores a single prediction state for each neighbor room. For that reason we chose the denotation 1-state context predictor (short: 1-state predictor). When leaving a room R the target room T is stored. When the person reenters room R, room T will be predicted as next room.

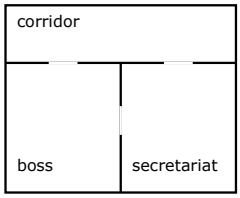


Fig. 1. Floor plan of corridor, boss' office, and secretariat

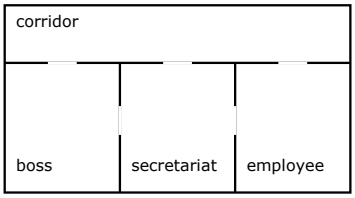


Fig. 2. Floor plan of corridor, boss' office, secretariat, and employee's office

The 1-state predictor works analogical to the one-bit branch predictor, if only two neighbor rooms exist. For example the corridor in figure 1 has only the secretariat S and the office of the boss B as neighbor rooms. Figure 3 shows the prediction graph of the 1-state predictor for the corridor. The states indicate which room will be predicted. If a person goes from the corridor into the boss'

office B the predictor will be set to the state B. When the person reenters the corridor, the boss' office B will be predicted as the person's next location. If this prediction proves as correct, the predictor stays in state B. Otherwise, if the person enters the secretariat S, the predictor changes to the state S and at the next time it will predict the secretariat.

Assuming three neighbor rooms the one-bit branch predictor has to be extended. If the corridor has the office of an employee E as an additional neighbor room (see figure 2), the prediction graph must be changed as in figure 4. The construction principle can be continued analogically in case of more neighbor rooms. Each room is represented by a node and all nodes are completely connected to each other.

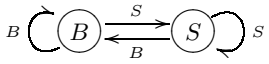


Fig. 3. Prediction graph of 1-state predictor for the corridor with two neighbor rooms

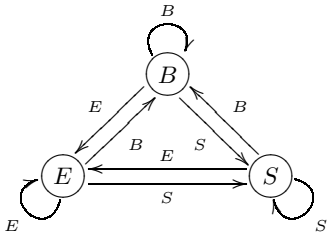


Fig. 4. Prediction graph of 1-state predictor for the corridor with three neighbor rooms

The storage costs of the 1-state predictor are very low. Only the id of the room that is predicted has to be stored for each room. An advantage of the 1-state predictor is its fast training. A person has to visit a room only once in the past and yet the next room can be predicted. A possible disadvantage is its potentially too fast retraining as the following example shows. A person goes always from the corridor into the same room. If she interrupts her habit only once by entering another room, the predictor is already retrained and predicts next time the wrong room. The 1-state predictor is very sensible against one-time deviations from the habit. Here the 2- or k-state predictor constitutes a remedy.

2-State Context Predictors. The 2-state context predictor is a modification of the two-bit branch predictor with saturation counter. The first entry denotes the next room as in the 1-state predictor. The second entry is used for changing between the strong and weak states. The room stored in the first entry is thus always predicted independently of the second entry, which influences training and retraining speed. The denotation "2-state context predictor" stems from the provision of two states for each predicted room.

At first let's assume again a room with two neighbor rooms (see figure 1). Figure 5 shows the corresponding prediction graph of the 2-state predictor. The denotations of the states consist of the id of the room and a counter. If a person

enters for the first time the boss's office B from the corridor, the state $B0$ is set. If the person reenters the corridor, the office of the boss B is predicted as next location. If the prediction proves as correct, the predictor switches into the strong state $B1$. Thus, next time the office of the boss B will be predicted again. If the person interrupts her habit once by entering the secretariat S, the state is set back from $B1$ to $B0$. Thus the boss' office is still predicted. Only if the person goes twice successively from the corridor into the secretariat S, the state is changed to $S0$ and next time the secretariat is predicted.

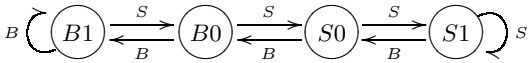


Fig. 5. Prediction graph of 2-state predictor for the corridor with two neighbor rooms

The prediction graph of a room with three neighbor rooms (see figure 2) is shown in figure 6. First training and transition to a strong state work as described above. If a person went several consecutive times from the corridor into the office of the boss, the predictor is in the strong state $B1$. If she goes next time from the corridor into a room different from the boss' office, the predictor changes into the state $B0$ predicting still the office of the boss. If the person goes now from the corridor into the secretariat the predictor switches into the state $S0$ independently of the room entered from the corridor before, and predicts thus the secretariat as next.

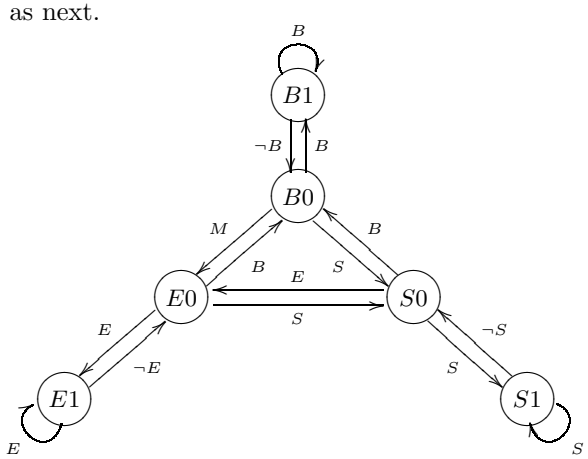


Fig. 6. Prediction graph of 2-state predictor for the corridor with three neighbor rooms

If a room has more than three neighbor rooms, the principle can be continued similarly. It should be noted that only the nodes with a 0 in the second entry, i.e. the weak states, form a completely connected graph as displayed in figure 6.

The storage costs of the 2-state predictor are still low. Besides the id of the room that is predicted an additional one-bit counter has to be stored for

each room. The computation costs for adapting the states are insignificantly larger than for the 1-state predictor. The predictor is also rapidly trained. The retraining is slowed down such that an one-time change of the habit does not cause an effect. In the case of two successive deviations from the habit the system notes the change. If more than two deviations of a habit should not yet lead to a retraining, the number of states must be increased leading to a k-state context predictor.

k-State Context Predictor. 1- and 2-state predictors are special cases of the k-state predictor, which features k states for each predicted room. The k-state predictor is related to the n-bit branch predictor, however $k = 2^{n-1}$. The first entry indicates again the next room. The further $n-1$ bits serve for enumerating the k states. If the counter is saturated, k mispredictions are needed before another room is predicted.

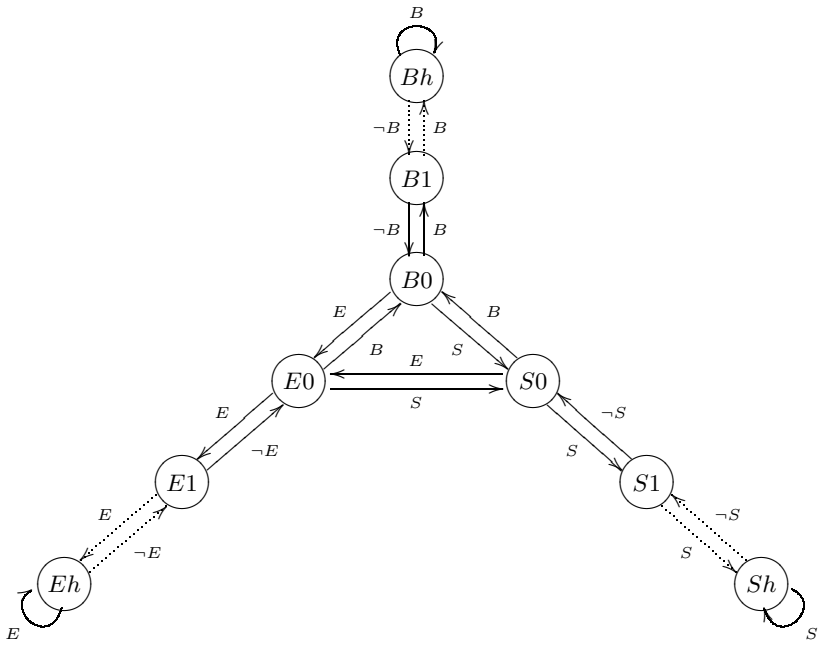


Fig. 7. Prediction graph of k-state predictor for the corridor with three neighbor rooms

The prediction graph (figure 7) shows a k-state predictor (with $h = k-1$) for a room with three neighbor rooms (see figure 2). The prediction graph shows that the k-state predictor is a generalization of the 2-state predictor. If the counter stands on $k-1$, the k-state predictor reaches its strongest states and k successive transitions into other neighbor rooms are needed until another room is predicted. The storage costs for the k-state predictor is not or only insignificantly larger than for the 2-state predictor.

The counter k indicates, how quickly a habit may be changed. An enhancement of the k-state predictor concerns the learning of k, i.e. the speed of a habit change could be learned.

Oscillating between two rooms leads to the fact that the wrong room is always predicted, if the 0 state is set for the first occurrence. Starting with a stronger state would halve the number of mispredictions. However, complex movement patterns cannot be learned by the 1-, 2-, or k-state predictors, but only by two-level predictors.

3.2 Global Two-level Context Predictors

The global two-level context predictors regard a sequence of the last rooms that a person entered to predict the next room. The visited rooms are stored in a kind of shift register that constitutes the first level of the predictor. If a new room is entered all entries of the register are shifted to the left and the new room is filled in from the right. The length of the shift register is called the order, which denotes the number of last visited rooms that influence the prediction. The second level consists of a pattern history table that stores all possible patterns of room sequences in different entries. Each entry holds additionally a 2-state predictor entry, which can be replaced by an arbitrary k-state predictor or a frequency analysis entry. The pattern in the shift register is used to select an entry in the pattern history table.

Second Level with 2-State Context Predictor. Here a 2-state predictor entry is used in the second level, i.e. a 2-state predictor that predicts the next room exists for each pattern in the pattern history table (see section 3.1).

We consider again the example with three rooms: C (corridor), S (secretariat), and B (office of the boss). Furthermore we assume an order of 3. Then there are $3 \cdot 2^2 = 12$ patterns and therefore 12 entries in the pattern history table. Figure 8 shows this case assuming the room sequence C S B C B S C S C B S C B S. After the first occurrence of the pattern C B S the initial state C0 is set for this pattern. The predictor changes to state C1 after the next occurrence of C B S. Now the prediction is that the corridor C will be entered next.

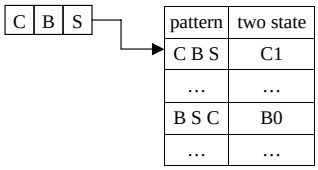


Fig. 8. Two-level predictor with 2-state predictor

The 2-state predictor in the second level can be replaced by a k-state predictor with different k to adjust the retraining speed.

Second Level with Frequency Analysis. The second level stores for each pattern the frequencies of all previous accesses to all neighbor rooms from the current room, which is listed last in the pattern. Now the room with the largest frequency is predicted. This predictor corresponds to the Markov Predictor from data compression [2].

We consider the same example as above. Figure 9 shows the shift register and the pattern history table for the two-level predictor with frequencies.

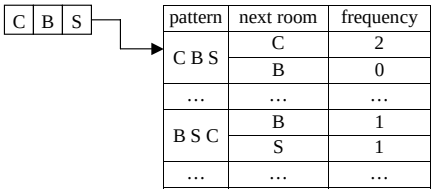


Fig. 9. Two-level predictor with frequencies

The storage costs depend on the number of rooms and on the selected order. Let n be the number of the rooms and let r be the order, then there are n^r patterns. However, sequences with two consecutively equal rooms (e.g. corridor - secretariat - secretariat) can never occur in case of the global two-level predictors described here. Therefore the number of possible patterns is $n \cdot (n-1)^{r-1}$, which constitutes the maximum storage size of the pattern history table assuming that each room is directly reachable from every other. In reality the complexity is further reduced, because not all rooms are neighbored.

If 2-state predictors are used in the second level, storage space is required for as many 2-state predictor entries as patterns. If the second level is realized with frequencies, the rooms, which are reachable from the last room of the pattern, together with the associated frequencies have to be stored for each pattern. Let $k_i (k_i < n,\ i = 1, \ldots, n)$ be the number of rooms, which are reachable from the i-th room, then the number of frequencies to be stored is $\sum_{i=1}^{n} (n-1)^{r-1} \cdot k_i$. For this number $n \cdot (n-1)^{r-1} \cdot (n-1) = n \cdot (n-1)^r$ is an upper limit.

An advantage of the global two-level predictors is that now complex movement patterns can be predicted. Since each pattern is treated separately, interferences [12] between two patterns cannot appear as is the case in branch prediction.

A disadvantages of the frequency analysis method in the second level is that after many turns of a pattern, retraining needs a long time. For example if a room R is entered 1,000 times after the same movement pattern, 1,000 times of entering another room after this pattern is needed before the prediction changes.

3.3 Local Two-level Context Predictors

The local two-level context predictors use only neighbor room history and disregard the movement sequences through all rooms. The shift register does not

contain the global history, i.e. which rooms were entered before, but the local history, meaning the rooms, which the person visited after the local room.

For each room exists a two-level predictor, i.e. the consecutively entered neighbor rooms form the pattern for each room. The local two-level predictors operate like the global two-level predictors. Thus again different prediction methods (2-state, k-state or frequency analysis) can be applied in the second level.

We look at the floor plan example of figure 2 assuming three neighbor rooms for the corridor. We assume a person moves through the rooms in the sequence C B C S B C B E C S C. The person is now in the corridor and the shift register used for the selection of the entry in the pattern history table of the corridor contains B S B S.

The storage costs are composed as follows. Let r be the order and let n_s be the number of the neighbor rooms of the room s, then a table with $(n_s)^r$ patterns has to be stored for each room s. The same room can occur successively in a pattern, in contrary to the global two-level predictors described above. An advantage over the global two-level predictors is that with a smaller order longer global sequences can be accounted for. Patterns of the kind "after repeated visits of a known room, another room is visited" can be recognized over a long sequence of movements. The disadvantage of this predictor is that the training phase is extremely long. Moreover, global movement patterns cannot always be distinguished. For instance the movement pattern C B C S B C B E C S C and C B C S E B E C B E C S C generate the same B S B S pattern of order 4, because only the rooms entered after corridor C are taken into account to form the local pattern. Thus, from the global view, interferences may occur in pattern construction.

The two-level context predictors can be extended using a method motivated by Prediction by Partial Matching (PPM) [2] from the area of data compression. Here a maximum order m is applied in the first stage instead of the fixed order. Then, starting with this maximum order m, a pattern is searched according to the last m rooms. If no pattern of the length m is found, the pattern of the length $m-1$ is looked for, i.e. the last $m-1$ rooms. This process can be accomplished until the order 0 is reached.

4 Evaluation

No benchmarks exist so far for movement patterns in ubiquitous systems. Therefore we use synthetic patterns for movements of a person within an office building.

First, we compare the 1-state, 2-state and k-state context predictors using simple movement sequences. We use the 4-state predictor as an example for the k-state predictor. Again we use the floor plan of figure 2 with the corridor, the boss' office B, the secretary S, and the employee's office E. The first column of table 1 shows the movement sequence and the other columns show the number of occurring mispredictions.

Table 1. Comparison of 1-state, 2-state, and 4-state predictors

	1-state	2-state	4-state
...B B B B B E B B B B B B B B B B B...	2 errors	**1 error**	**1 error**
...B B B B B E E B B B B B B B B B B...	**2 errors**	3 errors	**2 errors**
...B B B B B E E E B B B B B B B B B ...	**2 errors**	4 errors	3 errors
...B B B B B E E E E E E E B B B B B...	**2 errors**	4 errors	8 errors
...B B B B B E B E B E B E B B B B B...	8 errors	**4 errors**	**4 errors**
...B B B B B E B B E B B E B B B B B...	6 errors	**3 errors**	**3 errors**
...B B B B B E E B B E E B B E E B B...	**6 errors**	9 errors	**6 errors**
...B B B B B E E B E E B E E B B B B...	**6 errors**	7 errors	7 errors

When the movement pattern switches only between long sequences of the same room, then the predictors with less states are advantageous, because these predictors adjust fast to habit changes. Furthermore, the 4-state predictor doesn't perform better than the 1-state or 2-state predictors for any of the sequences.

Second, we describe simulation results evaluating all context predictors using various movement sequences as workload. Each sequence consists of a movement cycle, which is repeated until the person performed 1,000 room changes. All movement sequences are based on the four room scenario of figure 2. An order of 4 is used for the local as well as the global two-level predictors. The measurements calculate the cumulative error of every predictor, i.e. the mispredictions are added up. A predictor learned a pattern if the gradient of the cumulative error graph is zero. Learning a pattern means that the misprediction rate converges to 0%. The misprediction rate can be calculated from the charts as the cumulative error after 1,000 predictions divided by 1,000. The predictor labels shown in the diagrams are defined in table 2.

Table 2. Abbreviations for the context predictors

L-1L-1S	local one-level 1-state predictor
L-1L-2S	local one-level 2-state predictor
L-2L-2S(4)	local two-level predictor with 2-state predictor and order 4
L-2L-F(4)	local two-level predictor with frequencies and order 4
G-2L-2S(4)	global two-level predictor with 2-state predictor and order 4
G-2L-F(4)	global two-level predictor with frequencies and order 4

Measurement I: A weekly cycle is simulated using the same action from Monday to Friday, differing on Saturday and Sunday. The simulation uses the following cycle:

C - S - C - S - C - S - C - S - C - S - C - B - C - B (short: C $\overset{5\times}{\rightleftharpoons}$ S $\rightarrow$ C $\overset{2\times}{\rightleftharpoons}$ B)

Figure 10 shows the simulation results. Mispredictions of the local predictors occur on the exit of the corridor. None of the predictors is able to learn the

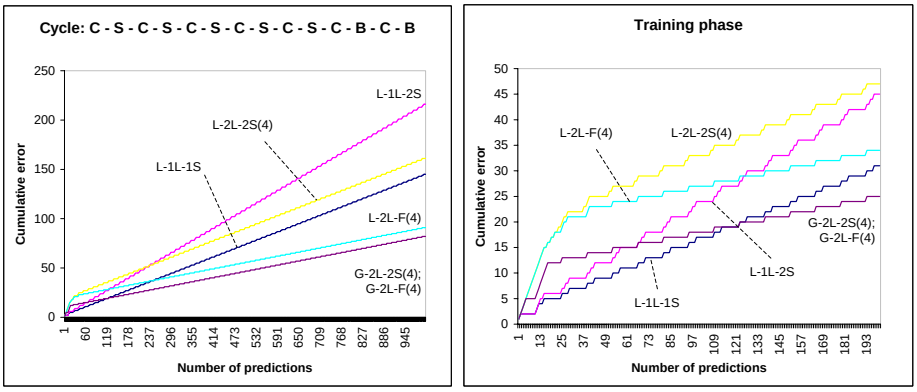


Fig. 10. Measurement I

pattern, because even the local two-level predictors with an order of 4 are not sufficient. An order of 5 would allow the local two-level predictors to learn this pattern. The 2-state predictor suffers most mispredictions (three per cycle), two at the change from S to B and one from B to S. The 1-state predictor generates one misprediction for each change between S and B, thus two per cycle.

The local two-level predictor with a 2-state predictor in the second level oscillates between S and B after the pattern S - S - S - S, consequently predicting the wrong room two times per cycle. This effect doesn't appear if frequency analysis is applied in the second level. The frequencies for S and B selected by the pattern S - S - S - S are equal after a complete cycle. But the predictor is implemented in a way to predict the room, that first reached that frequency, resulting in only one misprediction per cycle. The local two-level predictor with a 2-state predictor in the second level is even worse than the 1-state predictor in the overall view, because of the mispredictions caused by the training phase.

The global two-level predictors with a 2-state predictor and the frequency analysis in the second level perform equal. Both predictors suffer one misprediction per cycle after the pattern S - C - S - C before entering room B. The global two-level predictors suffer from less mispredictions than the local two-level predictors with frequency analysis because the global patterns of the training phase are proceeded faster than the local patterns.

Measurement II: Again we simulate a weekly cycle, however, with the same action from Monday to Thursday and a different action on Friday to Sunday resulting in the simulation cycle:

C - S - C - S - C - S - C - S - C - B - C - B - C - B (short: C $\overset{4\times}{\rightleftharpoons}$ S $\rightarrow$ C $\overset{3\times}{\rightleftharpoons}$ B)

Figure 11 shows the simulation results. The 2-state predictor makes four mispredictions per cycle, because it is in its strong state while changing from S and B. The 1-state predictor makes two mispredictions per cycle, because of the two changes per cycle in the pattern. The global two-level predictor with

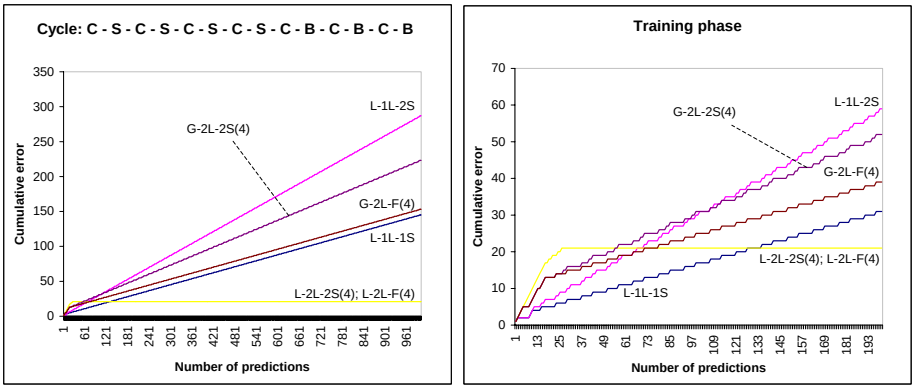


Fig. 11. Measurement II

frequency analysis suffers from two mispredictions per cycle after the patterns S - C - S - C and B - C - B - C. The global two-level predictor with 2-state predictor is in its strong state after the pattern S - C - S - C and before entering room B. Thus it predicts the wrong room and changes to the weak state of S. At the next occurrence of that pattern the predictor correctly predicts S and changes back to its strong state. After the pattern B - C - B - C the 2-state predictor alternates between the weak states of S and B, thus making two wrong predictions. Overall it performs three mispredictions per cycle.

The local two-level predictors learn the pattern after a training phase, making more mispredictions at that time as the other predictors. Measurements I and II demonstrate the rule that the length of the shift register must be at least the size of the iterations contained in the patterns.

Measurements III and IV: The patterns of the weekly cycle should be shortened now. The first cycle (measurement III) that should be analyzed is:

C - S - C - S - C - S - C - S - C - B (short: $C \overset{4\times}{\rightleftharpoons} S \rightarrow C \overset{1\times}{\rightleftharpoons} B$)

The second cycle (measurement IV) for the simulation is:

C - S - C - S - C - B (short: $C \overset{2\times}{\rightleftharpoons} S \rightarrow C \overset{1\times}{\rightleftharpoons} B$)

The local two-level predictors are able to learn the pattern in figure 12 left chart (measurement III). The global two-level predictors and the local one-level 2-state predictor make one misprediction per cycle at the change from S to B. The 1-state predictor predicts the wrong room two times considering the changes from S to B.

In figure 12 right chart (measurement IV) the local two-level predictors and the global two-level predictors learn the pattern, as it can be grasped by an order of 4. The difference is again in the training phase. The local one-level predictors perform as described earlier.

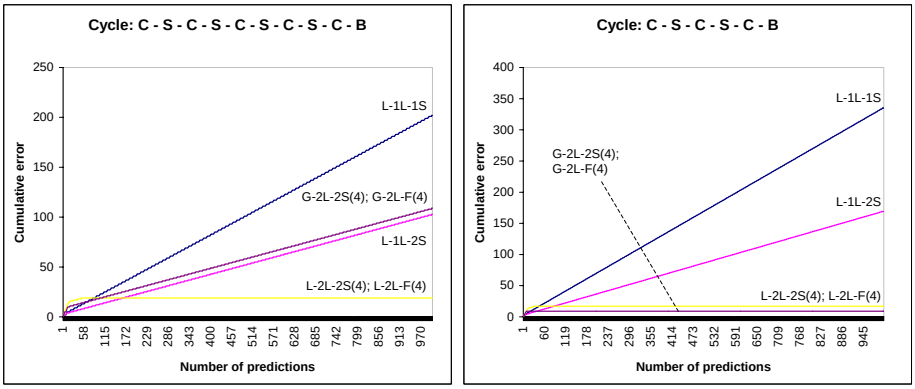


Fig. 12. Measurement III and IV

Measurements V to VII: These measurements examine the predictor's behavior regarding changing habits. The change can be that a person always carries out a single special action after a certain sequence of actions. Three different cycles V, VI and VII are simulated:

V: Z1 - Z1 - Z1 - Z1 - Z2 - Z2 - Z2 (short: 4×Z1 - 3×Z2)
VI: Z1 - Z1 - Z1 - Z1 - Z1 - Z2 - Z2 - Z2 - Z2 (short: 5×Z1 - 4×Z2)
VII: Z1 - Z1 - Z1 - Z1 - Z1 - Z1 - Z2 - Z2 - Z2 - Z2 - Z2 (short: 6×Z1 - 5×Z2)
Auxiliary cycle Z1: C $\overset{4\times}{\rightleftharpoons}$ S $\rightarrow$ C $\overset{1\times}{\rightleftharpoons}$ B
Auxiliary cycle Z2: C $\overset{4\times}{\rightleftharpoons}$ S $\rightarrow$ C $\overset{1\times}{\rightleftharpoons}$ E

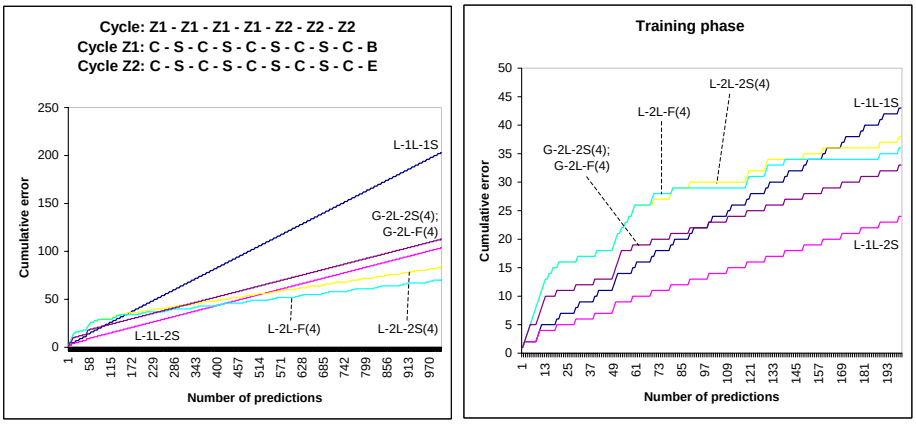


Fig. 13. Measurement V

Figures 13, 14, and 15 show that in all simulations the local one-level predictors behave equal for the retraining. They learn the changed habit after 1,

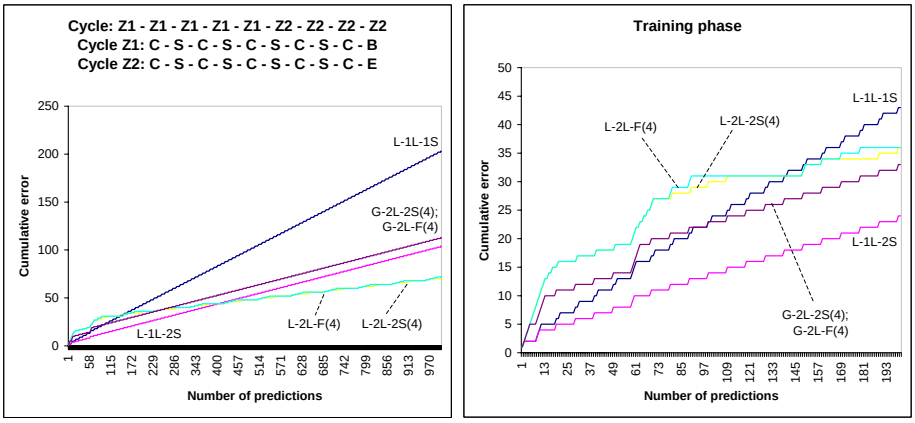


Fig. 14. Measurement VI

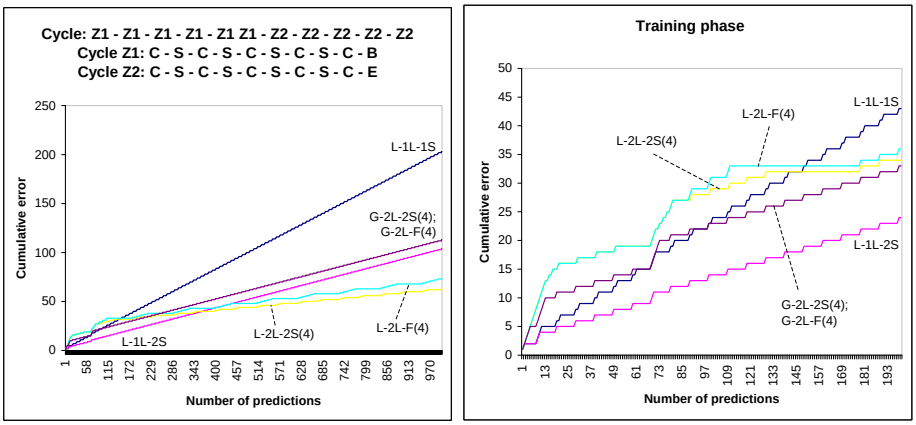


Fig. 15. Measurement VII

respectively 2 changes, not regarding the history. Also the global two-level predictors don't show differences in the simulation results, because the patterns are to long at a global point of view. That's why only the local two-level predictors will be investigated in more detail.

In measurement V, the local two-level predictor with frequency analysis makes less prediction errors. Because of the four changes to B after the local pattern S - S - S - S, it predicts three times the wrong room, if there is a change to S. The local two-level predictor with 2-state predictor makes an error at the change to B and E after the pattern S - S - S - S, i.e. it predicts four times the wrong room in each complete cycle.

The predictor with frequency analysis behaves equal in measurement VI, meaning that it always makes a mistake if there is a change to B. Therefore

it predicts the wrong room four times per cycle. The predictor with 2-state predictor in second level makes also four errors per cycle. The difference can be identified during the first retraining phase. The cumulative error after two transitions in the retraining phase is less for the 2-state predictor than for the predictors with frequency analysis. But as the 2-state predictor must relearn the transition to S, the cumulative error adjusts during the first transitions to S in each cycle to that of the predictor with frequency analysis, which doesn't make any mistake at transitions to S.

In measurement VII the 2-state predictor performs better than the predictor with frequency analysis. The 2-state predictor makes four errors per cycle, while the predictor with frequency analysis makes five errors. In the training phase both behave equivalent, but a noticeable difference can be spotted at the first retraining phase.

The simulation could be continued using this schema. The predictor with a 2-state predictor in the second level would make four mispredictions per cycle, while the misprediction rate of the predictor with frequency analysis in the second level would increase with the number of simulated auxiliary cycles.

Measurement VIII: This simulation analyzes random action sequences. The following cycle will be used:

C - {S,B,E}

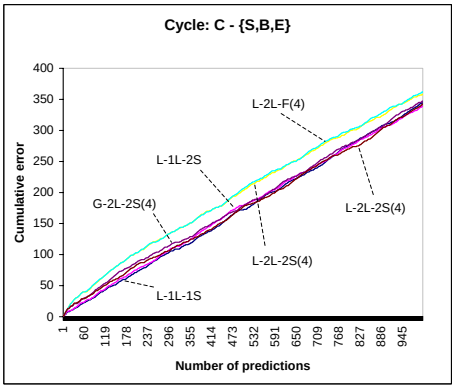


Fig. 16. Measurement VIII

If the transitions are selected randomly (figure 16), all predictors suffer from a comparable number of mispredictions. The local one-level predictors make a little less prediction errors as the global two-level predictors, which again are insignificantly better than local two-level predictors. The training phase and the relearning phases are the cause for this behavior.

5 Conclusion

This paper analyzed the suitability of branch prediction techniques known from the area of processor architecture for the context prediction. We transferred branch prediction techniques to context prediction using a person's movement patterns in a building. The evaluation of the implemented predictors used synthetic movement sequences, because of the lack of real movement pattern. The usage of various synthetic pattern lead to a good differentiation between the predictors, which are summarized as follows:

1-State Context Predictor. The 1-state predictor makes less prediction errors for long constant cycles than for cyclic patterns, as expected, which means that the same room is entered from another room many times. The 1-state predictor shows the best results for the training phase, because it only needs one transition from the actual room and no further history. The retraining performs also very fast. If the following room is changed, it will predict that room already for the next prediction request. The fast relearning is advantageous regarding long and constant cycles. In that case the 1-state predictor performs better than any other predictor (see figure 10 and 11).

2-State Context Predictor. Concerning long constant cycles (figure 10 and 11) the 2-state predictor performs worse than the 1-state predictor, because of its technique that a prediction must miss twice before it is changed. For cycles containing only one time a different successor (figure 12) the 2-state predictor performs better than the 1-state predictor with only one misprediction per cycle.

The training phase of the 2-state predictor works as fast as that of the 1-state predictor. A prediction can already be performed after one past transition. Retraining is slightly slower as of the 1-state predictor, as we need two misprediction to retrain. The 2-state predictor is suitable as a "warming-up predictor" for complex predictors.

The local one-level predictors, i.e. the 1-, 2, and k-state predictors, are unable to learn complex non-constant patterns. The state number k determines the retraining speed.

Local Two-level Context Predictors. These predictors cannot learn the pattern shown in figure 10 with an order of 4, because of the 5 transitions from the corridor to the same room. The predictor can learn a pattern only if the local pattern exhibits a transition length that is less or equal than the order (see figure 11).

A disadvantage of the 2-state predictors in the second level is the oscillation between two weak states (figure 10). The predictor with frequency analysis prevents this behavior. But on the other hand this kind of predictors need long retraining processes as shown in figure 13, 14, and 15. Considering only two possible neighbor rooms, the predictor with frequency analysis is the limit for k-state predictors $k \rightarrow \infty$.

The training phase for the local two-level predictors takes the longest time compared to all other regarded predictors, as each room must be left at least as often as the order to make a prediction. Retraining may concern two cases:

First, if a new pattern is used, a room must be left as often as the order specifies. Second, retraining needed within a complex pattern depends on the second level. Here, the 2-state predictor performs well, in contrast to the predictor with frequency analysis, which shows some weaknesses with the patterns in figure 15.

Global Two-level Context Predictors. In contrast to the local two-level predictors, the global two-level predictors don't learn the pattern in all measurement cases, as the following room must not change for a global pattern to be learned (figure 12 right chart). The patterns in the other figures can be learned with an order number of 5 or more.

The differences between the 2-state predictor and the frequency analysis in the second level is equivalent to those of the local two-level predictors. Figure 11 demonstrates the oscillation between two weak states.

In the training phase the global two-level predictors perform always better than the local once, because global patterns are learned faster. However, overall performance of the global two-level predictors is worse than the performance of the local one-level predictors. An analogical behavior is reflected at the retraining phase in the case of new patterns. In all other cases the relearning of a room, following a certain pattern, depends on the second level, equivalent to the local two-level predictors.

The simulations show that the 1- and 2-state predictors are very fast in training and retraining, whereas the two-level predictors are better suited for complex patterns. The advantages of both types could be combined in a hybrid context predictor. As an example, a 2-state predictor could be used during the training phase of a two-level predictor and its prediction is substituted by the two-level predictor afterwards. Such a hybrid predictor might be suitable for real world applications, as nobody wants to wait a long time for a system to adopt itself.

To avoid misguidance of persons or systems with wrong predictions, the confidence of the predictions should be taken into account. Meaning that a prediction should only be made, if the prediction reaches a high confidence level and suppressed otherwise.

Our future work concerns construction of new predictors and evaluation of these and of the described predictors with real movement sequences. A person tracking system, currently build up at the University of Augsburg, will generate such real movement patterns. Time is another important point in learning human habits. Therefore the predictors shall be enhanced to be time-dependent.

References

1. Ehrhard Behrends. *Introduction to Marcov Chains.* Vieweg, 1999.
2. I-Cheng K. Chen, John T. Coffey, and Trevor N. Mudge. Analysis of Branch Prediction via Data Compression. In *ASPLOS VII*, pages 128–137, Cambridge, Massachusetts, USA, October 1996.

3. Hans-W. Gellersen, Albrecht Schmidt, and Michael Beigl. Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts. In *Mobile Networks and Applications 7*, pages 341–351, 2002.
4. Kevin Gurney. *An Introduction to Neural Networks*. Routledge, 2002.
5. J. L. Hennessy and D. A. Patterson. *Computer Architecture a Quantitative Approach*. Morgan Kaufmann, San Mateo, CA, 2nd edition, 1996.
6. Finn V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, 1996.
7. Khomkrit Kaowthumrong, John Lebsack, and Richard Han. Automated Selection of the Active Device in Interactive Multi-Device Smart Spaces. In *Workshop at UbiComp'02: Supporting Spontaneous Interaction in Ubiquitous Computing Settings*, Göteborg, Sweden, 2002.
8. Eleftheria Katsiri. Principles of Context Inference. In *Adjunct Proceedings UbiComp'02*, Göteborg, Sweden, 2002.
9. Michael C. Mozer. The Neural Network House: An Environment that Adapts to its Inhabitants. In *AAAI Spring Symposium on Intelligent Environments*, pages 110–114, Menlo Park, CA, 1998.
10. S.-T. Pan, K. So, and J. T. Rahmeh. Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation. In *ASPLOS V*, pages 76–84, Boston, USA, April 1992.
11. Wolfgang Trumler, Faruk Bagci, Jan Petzold, and Theo Ungerer. Smart Doorplate. In *First International Conference on Appliance Design (1AD)*, Bristol, GB, May 2003.
12. J. Šilc, B. Robič, and T. Ungerer. *Processor Architecture - From Dataflow to Superscalar and Beyond*. Springer-Verlag, Berlin, Heidelberg, New York, 1999.
13. T.-Y. Yeh and Y. N. Patt. Alternative Implementation of Two-Level Adaptive Branch Prediction. In *Proceedings of the 19th Annual Symposium on Computer Architecture (ISCA-19)*, pages 124–134, Gold Coast, Australia, May 1992.
14. T.-Y. Yeh and Y. N. Patt. A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History. In *Proceedings of the 20th Annual International Symposium on Computer Architecture (ISCA-20)*, pages 257–266, San Diego, CA, May 1993.