UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK

LEHRSTUHL VIII
KÜNSTLICHE INTELLIGENZ

Inferring Probabilistic Automata from Sensor Data for Robot Navigation

LS–8 Report 18

Anke Rieger

Dortmund, May 22, 1995

Universität Dortmund
Fachbereich Informatik

University of Dortmund
Computer Science Department

Forschungsberichte des Lehrstuhls VIII (KI)
Fachbereich Informatik
der Universität Dortmund

ISSN 0943-4135

Anforderungen an:

Universität Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

Research Reports of the unit no. VIII (AI)
Computer Science Department
of the University of Dortmund

ISSN 0943-4135

Requests to:

University of Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

e-mail: reports@ls8.informatik.uni-dortmund.de
ftp: ftp-ai.informatik.uni-dortmund.de:pub/Reports
www: http://www-ai.informatik.uni-dortmund.de/ls8-reports.html

Inferring Probabilistic Automata from Sensor Data for Robot Navigation

LS–8 Report 18

Anke Rieger

Dortmund, May 22, 1995

Universität Dortmund
Fachbereich Informatik

Abstract

We address the problem of guiding a robot in such a way, that it can decide, based on perceived sensor data, which future actions to choose, in order to reach a goal. In order to realize this guidance, the robot has access to a (probabilistic) automaton (PA), whose final states represent concepts, which have to be recognized in order to verify, that a goal has been achieved. The contribution of this work is to learn these PA's from classified sensor data of robot traces through known environments. Within this framework, we account for the uncertainties arising from ambiguous perceptions. We introduce a knowledge structure, called prefix tree, in which the sample data, represented as cases, is organized. The prefix tree is used to derive and estimate the parameters of deterministic, as well as probabilistic automata models, which reflect the inherent knowledge, implicit in the data, and which are used for recognition in a restricted first-order logic framework.

(This paper is also published in M. Kaiser (ed.), Proceedings of the Third European Workshop on Learning Robots, 1995.)

1 Introduction

One problem in navigation is to guide a robot in such a way, that it can choose future actions, in order to achieve a goal, by taking into account perceived sensor data. Sensor measurements are not only to be used for verifying goals, but also for reacting to unexpected situations. In our approach we realize this guidance by giving the robot access to a (probabilistic) automaton, whose final states represent concepts, which have to be recognized, in order to verify, that a goal has been achieved. The contribution of this work is to learn this automaton from classified sensor data of robot traces through known environments.

In [10] and [11], operational concepts have been introduced, which are used by a human user to guide a robot. They constitute the basis for high-level planning, but are also symbolically grounded in robot perceptions. Rules have been learned, which derive these abstract concepts in several steps from sensor data and robot actions. The work presented in this paper complements the learning approach taken in [10], [11] in the following way: The learned rules, which are represented in a restricted first-order logic framework, do not take into account the uncertainties, which are associated with sensor observations: rules were learned, which derive different concepts from the same sensor data. This means, that the robot cannot distinguish two different states, for which it might be necessary to be recognizable as being different, because they require different actions. Given two rules with different conclusions, but the same premise, and given a situation, in which the premise is satisfied, the robot is not sure, in which state it is and it has difficulties in choosing the next action. In that case probabilities offer a solution in the following sense: If we can associate probabilities with the ambiguous rules, the robot can assume the object, which is most probable, in the first place, and can select the next action according to this assumption.

The aim of this work is to estimate these probabilities. With this goal in mind, we organize the data, from which the automata are to be learned, in a *prefix tree*, which can be mapped to (probabilistic) automata models, which reflect the inherent structure of the knowledge, implicit in the data. We consider deterministic and non-deterministic finite state automata (DFA's and NFA's), and hidden Markov models (HMM's). Each of these models has already been applied to robotics . In addition, there exist already several learning algorithms, which try to determine or estimate the parameters of these models (e.g., [1],[9],[13],[5],[12]). The characteristic feature of these existing approaches is, that they work on representations in propositional logic. Therefore, these learning algorithms cannot be applied directly to the domain used in [11], which is represented within a restricted first-order logic. The existing algorithms learn the structure of the automaton and estimate the probabilities of transitions, respectively. We show, how we can achieve the same objectives for a more complex representation by constructing the prefix tree, and by deriving relative frequencies from the data associated with it. These relative frequencies are taken as estimates for transition probabilities. In addition, we show, how the data, associated with the prefix tree, can be used for the generation of *predictions* and *foci of attention*.

We proceed as follows: In Section 2, we present the approach in the context of learning operational concepts. In Section 3, we show, how the training data has to be prepared, in order to derive the automata models. Section 4 deals with the generation and use of

DFA's for recognition. Section 5 addresses the ambiguities, i.e., PA's. In Section 6, we present first results of experiments in the robotics domain. Section 7 deals with prediction and focus of attention. We conclude in Section 8 with a discussion of related and future work.

2 The Approach

In [10], operational concepts were developed, which are used by a robot to perform a user-defined task in a flexible way. On one hand, operational concepts provide the basis for high-level planning. On the other hand, they are symbolically grounded, in the sense, that they can be traced down to basic robot actions and perceivable observations. Operational concepts constitute the highest level of the abstraction hierarchy presented in Figure 1. At the lowest level, we have the real-world data containing information about

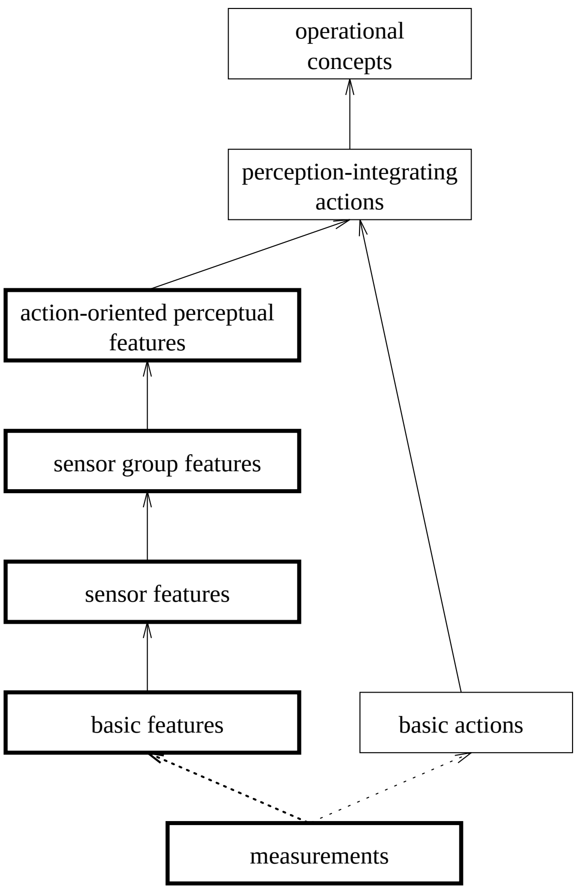

Figure 1: Abstraction hierarchy

sonar sensor measurements and robot positions. As we cannot bridge the gap between the real-world data and the operational concepts, such as `move through door` in one step, we introduced intermediate levels of abstraction. The strategy is then to learn off-line, how to derive higher-level concepts from lower-level ones. The learning steps are indicated in Figure 1 by the directed non-dashed arcs. In this paper, we are primarily concerned with the recognition of concepts from observations, and therefore focus on the left side of the hierarchy. In our learning scenario, we used data of robot traces through known environments, such as the one in Figure 2. An example for an *action-oriented perceptual feature* is the concept of moving through a doorway. With reference to Figure 2, we illustrate, what happens during robot traces, in which the robot moves through a doorway: The sensors on the robot's right side will first perceive the doorframe, labelled `9`, and then the wall, labelled `7`. Correspondingly, the sensors on the robot's left side will perceive wall `3` and `5`. In [11], we introduced the term `jump` for these kind of edge groupings, consisting of two parallel edges. An example of a rule [1], which has been learned by ILP-algorithms is the following:

```
through_door(Trace,Start,End,parallel) <-
  sg_jump(Trace,left,T1,T2,parallel) &
  sg_jump(Trace,right,T1,T2,parallel) &
  Start < T1 & T2 < End.
```

It states, that the robot moved `parallel`y through a doorway in a `Trace` during the interval frome time point `Start` to `End`, if, during a subinterval, the sensors on the robot's `right` and `left` side perceived the edge grouping `jump`. Other edge groupings, which are considered, are `convex` and `concave` corners, and singular edges, called `line`. *Sensor group features* are defined in terms of *sensor features*. Both describe the same event, namely the perception of an edge grouping during a time interval, during which the robot moved in a relative orientation towards, along, or away from the grouping. As the name already indicates, sensor group features describe the event for a group of sensors, whereas sensor features provide the information for a single sensor. Sensor group features are derived, if sufficiently many sensors, which are adjacent and belong to the same class, have perceived the same edge grouping:

```
sg_jump(Trace,right,TS,TE,parallel) <-
  s_jump(Trace,Sensor1,TS,TE,parallel) &
  s_jump(Trace,Sensor2,TS,TE,parallel) &
  adjacent(Sensor1,Sensor2) &
  sclass(Trace,Sensor1,T1,T2,right) &
  sclass(Trace,Sensor2,T1,T2,right) &
  T1 < TS & End < TE.
```

This rule states, that the sensors at the robot's `right` side perceive a `jump` during the time interval form `TS` to `TE` during which the robot moves `parallel`y along it, if at least two sensors, which belong to the class `right` perceived this grouping. Assume the situation, that `t12` denotes one of the traces in Figure 2, in which the robot moves parallely through the doorway. In this situation sensor `s5` on the robot's right side perceives during the time interval from time point `1` to `10` the `jump`, consisting of walls `9` and `7`. The robot moves `parallel`y along this grouping. This situation is described by the predicate instance

```
s_jump(t12,s5,1,10,parallel).
```

[1]We use a Prolog-like notation, i.e., variables begin with capital letters, constants with small letters.

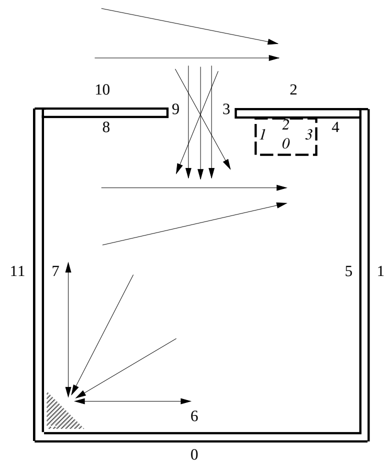

Figure 2: Room with robot traces

While moving, sensor `s5` receives a sequence of sonar sensor measurements, illustrated in Figure 3. They are grouped together in time intervals, during which the tendency of change of the measurements remains approximately the same (for details see [10],[17]). These time intervals are described by the *basic features* on the right side of Figure 3. The

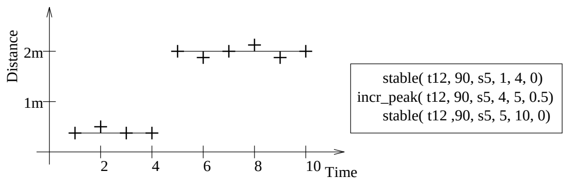

Figure 3: Sequence of sensor measurements

first one states, that in trace `t12` sensor `s5` received during the time interval from time point `1` to `4` approximately `stable` measurements. During this time interval the robot perceived the doorframe, labelled `9`. The stable measurements during the time interval from `5` to `10` correspond to the wall `7`. The measurements at time points `4` and `5` differ significantly, which is reflected by the `incr_peak` predicate. (The second argument of the basic features denotes the orientation of the sensors. It is taken into account in order to ensure, that the sensor orientation does not change. The last argument denotes the average gradient, which was measured during the time interval. It thus adds more details to the classification `decreasing`, `increasing` etc.). An example of a rule, which derives sensor features from basic features, and which has been learned within the ILP-framework, is the following:

```
s_jump(Trace,Sensor,T1,T4,parallel) <-
    stable(Trace,Or,Sensor,T1,T2,Grad1) &
```

```
incr_peak(Trace,Or,Sensor,T2,T3,Grad2) &
  stable(Trace,Or,Sensor,T3,T4,Grad3).
```

We will illustrate the inference of automata with this learning step, i.e., learning, how to derive sensor features from basic features, taking into account ambiguities. We start with a set of examples, E, and background knowledge B. In our application, both sets consist of ground literals. The predicates occurring in E are called *target predicates*, which may appear in the conclusion of a rule. The predicates in B are called *defining predicates*, which may appear in the premise of a rule. In the context of automata, instances of defining predicates constitute the sequences, which are input to the automaton, whose final states are associated with target predicates, representing the concept(s), which have been recognized. In the following, a target predicate of our domain, i.e., a sensor feature predicate, is denoted by `sf` $\in$ `SF={s_line, s_concave, s_convex,s_jump}`. A defining predicate, i.e., a basic feature, is denoted by `bf` $\in$ `BF={increasing,decreasing,stable, no_measurement, incr_peak,...}`. The goal is to infer from B and E an automaton, which takes as input sequences of basic features $\mathtt{bf}^1 \ldots \mathtt{bf}^k$ in temporal order, which eventually lead to an accepting state, tagged by sensor feature(s) in `SF`. The method for inferring (probabilistic) automata consists of the following steps:

1. Represent the training data as *cases*;
2. given the cases, construct the prefix tree;
3. derive the DFA from the prefix tree.

If the analysis of the DFA yields, that there are significantly many ambiguous states,

4. transform the DFA to a NFA;
5. evaluate the prefix tree, in order to estimate the transition probabilities.

The inferred (probabilistic) automaton is used for recognition, taking as input sequences of basic features. If a sequence leads to a final state, the recognized sensor feature(s) will be output and propagated up the abstraction hierarchy for further processing, enabling the robot to react to its observations.

3 Data Preparation

Cases Given the sets E and B, the data has to be prepared in such a way, that each sensor feature predicate instance is associated with the sequence of relevant basic features. This task is accomplished by generating a case for each example in E. A *case* is represented by a list `[target instance|defining instances]`, which contains all the defining instances, which are *relevant* for the respective target instance. In our application instances of basic features are *relevant* for a target instance, if they refer to the same trace, to the same sensor, and to the same time interval, respectively. In addition we ensure, for our domain, that the defining instances are sorted according to temporal order. Thus, the sequences of defining instances in the set of cases constitute the input sequences, which have to be accepted by the inferred automaton.

Prefix Tree Given a set of cases, each of which consists of an instance of a target predicate and a sequence of instances of defining predicates, which in our domain describe a temporal process, we organize this sample data in a *prefix tree*, such that a sequence of defining predicates corresponds to a path from the root node to another node of the tree, which contains the information about the corresponding target instance. The nodes are associated with the following information (see Figure 4):

- *Label*: label of the node (sequence of defining predicates);
- $\#CC$: number of cases completely covered by the node;
- $\#SC$: number of cases covered by the subtree, whose root the node is;
- CC: cases, which are completely covered by the node
- SC: cases, which are covered by the nodes of the subtree, whose root the node is.

The edges are labeled by a defining predicate, in our domain a basic feature. A case is *covered* by a node, if its sequence of defining predicates corresponds to the labels of the edges of the path from the root to the respective node. Given a node i with label $\mathtt{bf}^1 \ldots \mathtt{bf}^k$, $CC(i)$ contains those cases, whose sequence of defining predicates matches exactly the label, whereas $SC(i)$ contains the cases, which have the label $Label(i)$ as prefix, i.e., whose sequences of defining predicates match a label $\mathtt{bf}^1 \ldots \mathtt{bf}^k \ldots \mathtt{bf}^n$, $n > k$. The information stored in the nodes is absolutely redundant, but facilitates the access to the data, and therefore supports fast evaluation operations.

4 Derivation of DFA's

Construction of the DFA The prefix tree can easily be transformed to a DFA, which reflects the inherent structure of the knowledge, implicit in the data. A DFA is defined by the tuple $(Q, \Sigma, \delta, q_0, F)$, where Q denotes a set of states, Σ denotes the input alphabet, $\delta : Q \times \Sigma \rightarrow Q$ denotes the transition function, q_0 denotes the starting state, and F denotes the set of final states. We generate the DFA from the prefix tree in the following way: The input alphabet consists of the predicates for basic features, i.e., $\Sigma = \{$ `increasing(_,_,_,_,_,_)`, `stable(_,_,_,_,_,_)`, $\ldots\}$. For each node in the prefix tree we establish a state $q \in Q$. The starting state q_0 will be the state corresponding to the root node. The edge information of the prefix tree is used to generate the transition function δ: If there exists an edge from node i to node j, labeled by a basic feature $\mathtt{bf} \in \Sigma$, then establish the transition $\delta(q_i, \mathtt{bf}) = q_j$, where q_i and q_j represent the states, established for nodes i and j respectively. Those states, which were established for nodes in the tree with $\#CC > 0$, become the final states of the DFA. The final states in F are tagged with the concepts, described by the target predicates of the cases, covered by the corresponding node. As our input alphabet consists of predicates instead of propositional constants, the application of the transition function requires a test for unifiability. Thus we can check, whether a sequence of observations $\mathtt{bf}^1 \ldots \mathtt{bf}^k$, i.e., a sequence of ground predicates for basic features, is accepted by the automaton, and, if that is the case, the tags of the accepting state indicate the concept(s), which have been recognized.

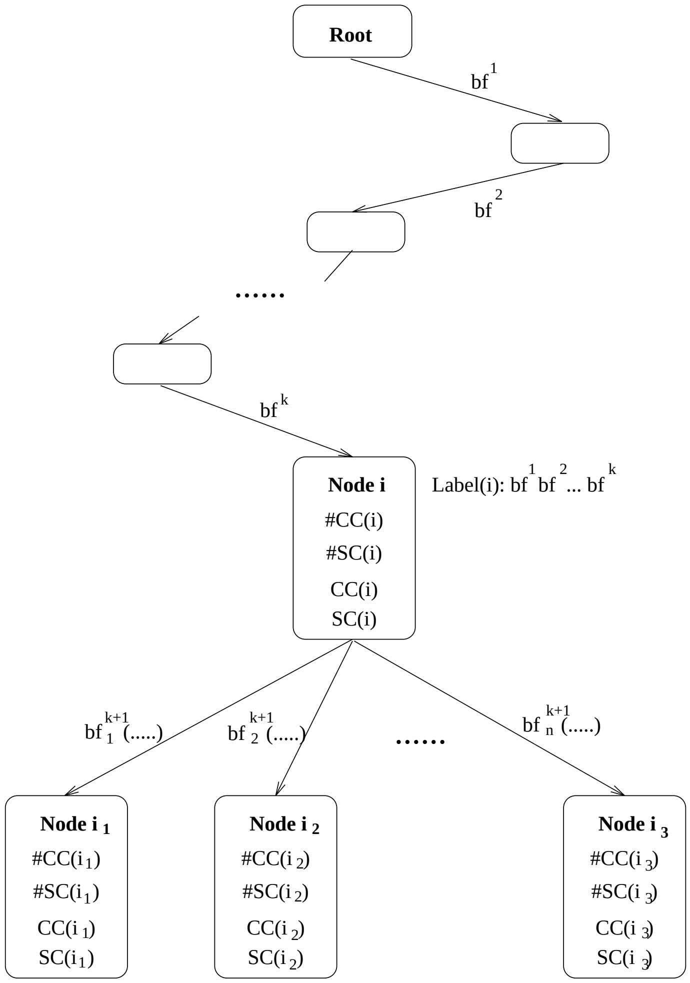

Figure 4: Prefix tree information

Using the DFA for object recognition When the robot moves through the environment, its sensors constantly perceive observations, each of which might be the beginning of a string, which is accepted by the automaton. Based on this idea, we can apply a simple *marker passing method* [7],[2] to the DFA. Assume, that the robot has perceived a sequence of observations $\texttt{bf}^1 \ldots \texttt{bf}^k$ during a period of time, in which no change of direction of movement took place. Then we can generate at each time point $t, 1 \leq t \leq k$ a marker m, which is associated, if possible, with the state of the DFA with $\delta(q_0, \texttt{bf})$, where the predicate $\texttt{bf} \in \Sigma$ is unifiable with the ground predicate $\texttt{bf}^t$. We denote the state, to which the marker $m_l, 1 \leq l \leq k$ is associated at time point t, by $q^t(m_l)$. Each of the markers $m_r, 1 \leq r < t$, which has been generated at previous time points, is passed from its previous state $q^{t-1}(m_r) = q_i$ to the next state $q^t(m_r) = q_j$, if there exists a transition $\delta(q_i, \texttt{bf}) = q_j$, such that $\texttt{bf}$ is unifiable with $\texttt{bf}^t$. Otherwise the marker is thrown away. If a marker has reached a final state, a message is produced, that an object has been recognized, which is possibly an instance of several concepts. So the algorithm can be sketched as follows:

Input: An observation sequence $\texttt{bf}^1 \ldots \texttt{bf}^k$.

Output: The concepts, instances of which have been recognized during subintervals of the time interval $[1, k]$.

Algorithm: For $t = 1, \ldots, k$

1. establish a marker m_t, and, if possible, associate it with the state $q = \delta(q_0, \texttt{bf})$, such that `bf` is unifiable with $\texttt{bf}^t$, i.e., $q^t(m_t) = q \in Q$;

2. for each marker $m_r, 1 \leq r < t$, with $q^{t-1}(m_r) = q_i$:
   if there exists a state q_j, such that $\delta(q_i, \texttt{bf}) = q_j$ and `bf` is unifiable with $\texttt{bf}^t$, then pass the marker to state q_j, i.e., $q^t(m_r) = q_j$; if $q^t(m_r) \in F$, return a message, that an instance of one or several concepts has been recognized; if the marker cannot be passed forward, throw it away.

5 Derivation of PA's

The problem We now turn to the problem of dealing with uncertainties, caused by ambiguous sensor observations. In order to evaluate the prefix tree with respect to the relative frequencies of ambiguities, we provide operations, which search for the nodes in the tree, which cover cases, whose instances of target predicates describe different concepts. Concepts in our domain can be described by predicates and combinations of predicates and argument values. Sensor features, e.g., `s_line(_,_,_,_,parallel)`, `s_line(_,_,_,_,straight_towards)`, and `s_line(_,_,_,_,straight_away)`, should be distinguishable for the following reason: if the higher-level goal is to *move along a wall*, this goal translates on the lower level to the sensor feature `s_line(_,_,_,_,parallel)`, which is different from `s_line(_,_,_,_,straight_towards)`, and `s_line(_,_,_,_,straight_away)`, because the latter would signal the failure of executing the operational concept of *moving along a wall*.

If we derive the DFA from a given prefix tree, situations, similar to the one sketched in Figure 5, may arise. The same sequence of basic features, namely `stable(_,_,_,_,_,_)`,

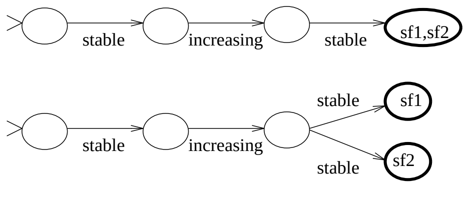

Figure 5: Ambiguous Perceptions

`increasing(_,_,_,_,_,_)`, `stable(_,_,_,_,_,_)`, leads to a final state, which is tagged with two different concepts, `sf1` and `sf2`. If these are `s_line(_,_,_,_,straight_towards)`

and `s_line(_,_,_,_,parallel)`, they should be distinguishable, because dependent on the higher-level goal, e.g., *moving along a wall*, they require different actions, namely a change of direction of the movement in the former case, and a continuation of the movement in the latter case.
Now the idea is to split up the final state into two, and to introduce non-deterministic state transitions (see Figure 5). The ultimate goal is to tag these non-deterministic transitions with probabilities, such that the robot can assume the concept, which is most probable, in the first place, and can choose its action accordingly.

Probabilistic automata (PA's) In order to model this situation, we switch to probabilistic automata models. If we want to be able to represent ambiguities by stochastic state transitions, the state transition function δ has to become a state transition probability distribution. One possibility is to use *Markov chains*, which are defined by the tuple (Q, Δ, π^1), where Q denotes a set of N states, Δ denotes a stochastic matrix containing N^2 transition probabilities, and π^1 is the initial state distribution. The transition probability δ_{ij} is defined as

$$\delta_{ij} = Pr(q_j^{t+1} | q_i^t) \; 1 \leq i, j, \leq N, \; t = 0, 1, 2, \ldots,$$

i.e., it is the probability that a process, which is in state q_i at time point t, will occupy state q_j after the next transition. Each δ_{ij} satisfies the condition $0 \leq \delta_{ij} \leq 1$, $1 \leq i, j \leq N$ Since the process must occupy one of its N states after each transition, we have $\sum_{j=1}^{N} \delta_{ij} = 1$, $i = 1, 2, \ldots, N$ The initial state distribution π^1 is defined as the vector $\pi^1 = \{\pi_i^1\}, 1 \leq i \leq N$, where

$$\pi_i^1 = Pr(q_i^1) \text{ with } 1 \leq i \leq N,$$

i.e., π_i^1 denotes the probability, that the system is in state q_i at time point 1.
Whereas Markov chains do not take into account observations, *hidden Markov models* do so. They are defined by the tuple $(Q, Z, \Delta, \Lambda, \pi^1)$, where Q, Δ, and π^1 are defined as described above, Z denotes the output alphabet (in our domain the basic features), and Λ denotes the observation probability distribution, which can be associated either with states (see [13]) or with state transitions (see [9]). In the latter case the distribution is defined as

$$\lambda_{ijk} = Pr(z_k^t | q_i^{t-1} \rightarrow q_j^t)$$
$$\text{with } 1 \leq i, j \leq |Q| \text{ and } 1 \leq k \leq |Z|,$$

i.e., λ_{ijk} denotes the probability, that the system observes the output symbol z_k at time point t, given that the transition from state q_i at time point $t-1$ to state q_j at the next time point t took place. The probabilities have to add up to one, i.e., the condition

$$\sum_{k=1}^{|Z|} \lambda_{ijk} = \sum_{k=1}^{|Z|} Pr(z_k^t | q_i^{t-1} \rightarrow q_j^t) = 1$$

has to be satisfied.
Whereas Markov chains are insufficient for our purposes, because of the missing observations, HMM's offer to much, as in our case the output function is deterministic, i.e., there exists a $k \in \{1, \ldots, |Z|\}$, such that $\lambda_{ijk} = 1$, and for all $l \neq k$ we have $\lambda_{ijk} = 0$. Nevertheless, we choose the HMM, because in future work we want to consider scenarios, in which we have to account for probabilistic outputs.

Prefix tree evaluation Before switching to the probabilistic framework, we have to evaluate our sample data with respect to the relative frequencies of ambiguities. Only if these frequencies are significant, it will be worthwhile to transform the DFA to a NFA and to estimate the transition probabilities of the non-deterministic transitions. Once, we have determined the nodes of the prefix tree, which cover cases, whose target predicate instances belong to different concepts, we can use the node information (see Figure 4) to determine the number and relative frequencies of the concepts, given that the corresponding sequence of observations has been perceived. Consider an arbitrary node i of the prefix tree, whose label is $\mathtt{bf}^1 \ldots \mathtt{bf}^k$. Let $c_1, c_2, \ldots, c_n$ denote the n concepts, which are represented by the cases covered by the node i, i.e., by the targets of the cases in $CC(i)$. Let $CC(i, c_j)$ and $\#CC(i, c_j)$ denote the cases of node i, which belong to concept $c_j, 1 \leq j \leq n$, and their number respectively. Then the relative frequency, that concept c_j has been recognized, given that the observation sequence $\mathtt{bf}^1 \ldots \mathtt{bf}^k$ has been observed is

$$RF\{c_j | \mathtt{bf}^1 \ldots \mathtt{bf}^k\} = \frac{\#CC(i, c_j)}{\#CC(i)} \tag{1}$$

Prefix tree → HMM Only if there are significantly many nodes in the tree, which are associated with different concepts, whose relative frequency is significantly high, it will be worthwile introducing probabilistic automata. This is done in the following way: We transform the prefix tree to a DFA as described above. Assume, that the evaluation of the prefix tree has yielded the result, that node j covers cases, whose target predicates describe several concepts, $c_1, \ldots, c_n$, which should be distinguishable, i.e., the state, corresponding to node j, q_j, is to be split up into n states, $q_{j_1}, \ldots, q_{j_n}$ (see Figure 6). Then, we have to introduce non-deterministic transitions from the state q_i corresponding to the predecessor i of node j, to the states $q_{j_1}, \ldots, q_{j_n}$. Furthermore, we have to add for each newly generated state $q_{j_1}, \ldots, q_{j_n}$ the deterministic transitions emanating from state q_j to its successors, which, of course, remain deterministic. In the second step, we have to

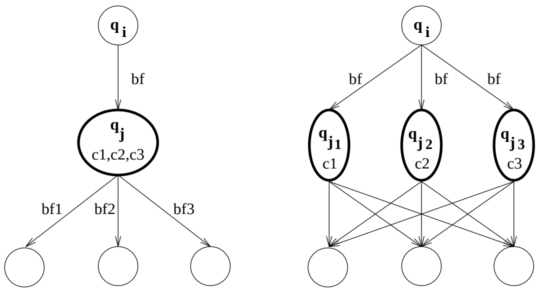

Figure 6: DFA → NFA

estimate the transition probabilities for the non-deterministic transitions from the sample data in the prefix tree. The goal is to estimate the transition probabilites $\delta(q_{j_l} | q_i), 1 \leq l < n$. Remember, that the index l identifies one of the concepts $c_1, \ldots, c_n$. Let $\mathtt{bf}^1 \ldots \mathtt{bf}^k$ be the label of node j. Then we can use Equation 1 to determine the estimators, i.e., we take the relative frequency, that concept c_l has been recognized, given, that $\mathtt{bf}^1 \ldots \mathtt{bf}^k$ has been perceived, as estimator for the transition probability from state q_i to state $q_{j_l}, l = 1, \ldots, n$:

$$\hat{\delta}_{ij_l} = \hat{Pr}(q_{j_l}^{t+1}|q_i^t) = \frac{\#CC(j, c_l)}{\#CC(j)} \tag{2}$$

As already mentioned, in our domain the observations associated with the transitions are deterministic, i.e.,

$$\hat{\lambda}_{zij_l} = \hat{Pr}(z^t|q_i^{t-1} \rightarrow q_{j_l}^t) = \begin{cases} 1 & \text{if } z = \texttt{bf} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

for $l = 1, \ldots, n$. The transitions from the newly introduced states $q_{j_1}, \ldots, q_{j_n}$ to the m successor states $q_{s_1}, \ldots, q_{s_m}$ of the original state q_j remain deterministic:

$$\delta_{j_l s_k} = Pr(q_{s_k}^{t+1}|q_{j_l}^t) = 1 \tag{4}$$

for $l = 1, \ldots, n$ and $k = 1, \ldots, m$. The deterministic output is determined in the same way as for the non-deterministic transitions (see Equation 3).

Use of the PA for object recognition Given the PA, derived from the prefix tree in the way described above, we can apply the *Viterbi algorithm* [13],[9]. This algorithm is an optimal search procedure for finding the most likely state sequence $\mathbf{q} = q^0 q^1 \ldots q^n$ of a Markov source, given an observation sequence $\texttt{z}=\texttt{bf}^1 \ldots \texttt{bf}^k$. In other words, it tries to find the sequence $\mathbf{q}^*$, that maximizes the conditional probability $P(\mathbf{z}|\mathbf{q})$. This algorithm can modified to work with our representation, as already illustrated for DFA's above. Then, given the estimate $\mathbf{q}^*$, we can investigate its last state q^n, whose tag represents the concept, which has been recognized most probably.

6 Experiments

In this section, we present first results of learning automata for the recognition of sensor features from basic features. For the experiments we used the data [2] of seven traces through a known environment illustrated in Figure 7. The experiments were set up in such a way, that they could give answers to the following questions:

1. What does the structure of the learned tree automata look like, referring to their depth, the number of states, and the number of final states, respectively?

2. How many of the final states are ambiguous?

3. How do different ways of calculating basic features effect the structure and the ambiguities?

The calculation of basic features can be guided by different parameters. One parameter is the tolerance, within which successive measurements and their gradients, respectively, are considered to be approximately equal (for details see [17]). This tolerance has been set to 6 (Version 1), 10 (Version 2), and 15 degree (Version 3), respectively. The effect of the different versions is illustrated with the sequence of measurements in Figure 8. For

[2] The data has been provided by the University of Karlsruhe

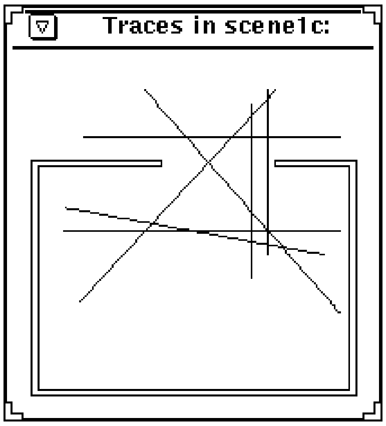

Figure 7: Scene with traces used for learning

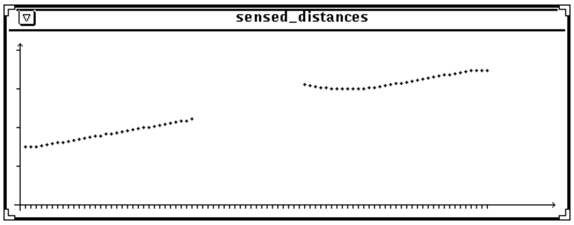

Figure 8: Sensor measurements

these measurements, each version calculated a different sequence of basic features:
Basic features for Version 1:

```
increasing(t76,75,s6,3,32,13).
no_measurement(t76,75,s6,32,53,999).
decreasing(t76,75,s6,53,59,-9).
stable(t76,75,s6,59,65,1).
increasing(t76,75,s6,65,69,11).
something_happened(t76,75,s6,69,70,18).
increasing(t76,75,s6,70,85,13).
```

Basic features for Version 2:

```
increasing(t76,75,s6,3,32,13).
no_measurement(t76,75,s6,32,53,999).
decreasing(t76,75,s6,53,63,-5).
increasing(t76,75,s6,63,85,12).
```

Basic features for Version 3:

```
increasing(t76,75,s6,3,32,13).
```

	Version 1	Version 2	Version 3
# cases	400	385	373
$\lvert Q\rvert$	225	146	108
$\lvert F\rvert$	143	100	75
Depth	9	7	6
$A1$	16	23	20
$A2$	25	25	22

Table 1: Results of learning

```
no_measurement(t76,75,s6,32,53,999).
stable(t76,75,s6,53,69,0).
increasing(t76,75,s6,69,85,13).
```

The first version is most sensitive to variances with the effect, that the sequence of time points is split up into smaller time intervals, resulting in the longest sequence of basic features. The difference between the second and third version again reflects the effect of the greater variance. The time region, which is labelled `decreasing` in Version 2 is considered `stable` under Version 3.

For each of the three versions, we organized the data in cases, constructed the prefix tree, and derived the corresponding NFA. For each automaton, we determined the number of states, $|Q|$, the number of final states $|F|$, and the depth of the tree, in oder to analyze the automaton structure. Then, we evaluated the final states with respect to ambiguities. In Section 5, we stated, that concepts in our domain can be characterized by the predicate (alternative $A1$) or by the predicate and the fifth argument (alternative $A2$). For each alternative, $A1$ and $A2$, we determined the number of final states, which were associated with more than one concept.

The results are summarized in Table 1. The different numbers of cases for each version are an effect of the way basic features and cases are generated. Above we have seen the tendency, that the bigger the tolerance is set, the bigger the time intervals for basic features become. Thus it may happen, that there is no basic feature instance, whose time interval is a subinterval of a given sensor feature example. So there might be examples, for which no cases can be found.

The greater the tolerance is chosen, the less states and final states the learned automata have. Furthermore, the depth of the trees decreases. This also reflects the effect, that the greater the tolerance for calculating basic features is set, the bigger the time intervals become, and the shorter the sequences of basic features.

We now turn to the ambiguities: Considering the differentversions for calculating basic features, the absolute number of final states, which are associated with different concepts, does not vary too much from version to version. However, if we consider the ratio between the number of ambiguous states and the number of final states, we get 17 % for Version 1, 25 % for Version 2, and 29 % for Version 3. Thus, the relative number of ambiguous states increases and becomes significantly high, the greater the tolerance for calculating basic features is set.

An example for an ambiguous state is the one, to which the sequence `incr_peak`, `stable` leads. For alternative $A2$, this state is associated with the concepts `s_jump(_,_,_,_,parallel)`, `s_convex(_,_,_,_,parallel)`, and `s_concave(_,_,_,_,parallel)`. The evaluation of the prefix tree yields for these concepts the relative frequencies 0.25, 0.5, and 0.25, respectively. The sequence `decreasing,stable,increasing` leads to a state, which is associated with the concepts `s_line(_,_,_,_,parallel)` and `s_line(_,_,_,_,diagonal)`. The relative frequencies for these concepts are 0.75 and 0.25, respectivley. They are used as estimates for the transition probabilities for the HMM, as explained in Section 5 (see Equation 2). So, given, that the latter sequence of basic features has been perceived, the Viterbi algorithm will output, that the concept `s_line(_,_,_,_,parallel)` has been recognized most probably, namely with probability 0.75.

7 Further Evaluations

In the previous sections, we were concerned with recognition and the interpretation of sensor observations, respectively. In this section, we show, how the information in the prefix tree can also be used to determine a focus of attention and to make predictions.

Focus of attention In this context, we address the situation, that the robot is to recognize a specific concept c. The robot has already observed a partial observation sequence $\mathtt{bf}^1 \ldots \mathtt{bf}^k$, which does not yield enough information, to derive, that an instance of concept c has been recognized. The question is, which future observation would yield the most evidence for it. Given the sequence $\mathtt{bf}^1 \ldots \mathtt{bf}^k$, we determine the node i of the prefix tree, which is reached, when following the path described by the sequence. With reference to Figure 4 let $i_1, \ldots, i_n$ denote the successor nodes of node i. Then we can calculate for each sucessor node $i_l, 1 \leq l \leq n$

$$RF\{c|\mathtt{bf}^1 \ldots \mathtt{bf}^k \mathtt{bf}_l^{k+1}\} \quad = \quad \frac{\#CC(i_l, c) + \#SC(i_l, c)}{\#CC(i) + \#SC(i)}, \tag{5}$$

which is the relative frequency, that the concept c will be recognized, given that the sequence $\mathtt{bf}^1 \ldots \mathtt{bf}^k \mathtt{bf}_l^{k+1}$ has been observed. The observation $\mathtt{bf}_l$, which will yield the most evidence for the concept c is

$$\max_{\{l \in 1 \ldots, n\}} \left\{ RF(c|\mathtt{bf}^1 \ldots \mathtt{bf}^k \mathtt{bf}_l^{k+1}) \right\}. \tag{6}$$

Prediction Recognition is concerned with the question: Given a sequence of observations, which concept *has been recognized* most probably? Prediction deals with the situation, that the robot tries to orient itself: Given a partial sequence of observations, which concept *will be recognized* most probably, when further observations are made? Given the sequence $\mathtt{bf}^1 \ldots \mathtt{bf}^k$, we determine the node i of the prefix tree, which is reached, when following the path described by the sequence. Let $c_1, \ldots, c_n$ denote the concepts associated with the cases in $SC(i)$. Then the concept, which will be recognized most probably, will be

$$\max_{j \in \{1, \ldots, n\}} \left\{ \frac{\#SC(i, c_j)}{\#SC(i)} \right\}. \tag{7}$$

8 Discussion

Inferring probabilistic automata requires, in the first step, the derivation of the automaton structure, i.e., the number of states, and the graph of possible transitions. Then, the probabilities of the transitions have to be estimated. We have organized the training data in a prefix tree, which was used to solve both tasks.

In principal, the first objective can also be achieved, by applying the L^*-algorithm, developed by Angluin [1]. The algorithm learns a regular set from a *minimal adequate teacher*, who is assumed to answer correctly membership queries and conjectures of the learner about the unknown regular set. There have been numerous approaches for infering automata, especially with applications to robotics, e.g. [12],[14],[16], and [5], which are based on L^*. Modifications of the algorithm seem to be possible, in order to apply it to our domain. Nevertheless, in our case, the construction of the prefix tree is a more straightforward way of attaining the automaton structure for a more complex representation.
Given the automaton structure, we have estimated the transition probabilities from the sample data associated with the prefix tree. An alternative would have been to apply *Viterbi-* or *forward-backward extraction* [9],[13], which are gradient descent methods to estimate the state transition and output probability distributions of a hidden Markov model. Given a set of output sequences $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_L\}$, where $\mathbf{z}_i = z_i^1 z_i^2 \ldots z_i^{n_i}$ with $i \in \{1, \ldots, L\}$, and a guess for the structure of the Markov source and its statistics, i.e., Q, $\hat{\delta}^{(0)}(q^t|q^{t-1})$ and, $\hat{\lambda}^{(0)}(z^t|q^{t-1} \to q^t)$, they determine estimates $\hat{\delta}$ and $\hat{\lambda}$ for the transition and output probability distributions, respectively. As in our domain of application it turned out, by evaluating the experiments, that most of the transitions and all outputs are deterministic, the calculation of the relative frequencies from the information associated with the prefix tree was a more direct way of acquiring the same objective. A topic of future research will be, to evaluate in more complex situations with probabilistic outputs, whether the application of one of the extraction methods will be more advantageous.

Other important related work stems from the field of machine learning and ILP. By constructing the prefix tree, we have extracted the knowledge structure, which is implicit in the sample data, and, ultimately, in the flat rule set, learned in [11]. When generating the knowledge structure, we use the temporal relation between events. However, the resulting hierarchy can also be interpreted as concept hierarchy with a generalization hierarchy. The internal nodes of the tree represent potentially useful concepts, which could be introduced, in order to support recognition and classification. From this point of view, our approach is related to concept formation and clustering. Future work will have to clarify the relation between this approach and those proposed for demand-driven concept formation [18], for rule restructuring [15], and for conceptual clustering [6].

We have shown, that it is possible to match sequences of observations onto input sequences of automata, whose final states indicate, that a concept, i.e., a sensor feature, has been recognized. Clearly, the work, presented in this paper, is work in progress, and future work will certainly include further experiments, tests, and evaluations. In principle, however, it is also possible, to design and learn automata for higher-level concepts, e.g., *perception-integrating actions* (see Figure 1). They associate sequences of observations of *action-oriented perceptual features* with sequences of *basic actions*, during which the features have been perceived. By mapping actions to the input alphabet of a DFA and observations to its output alphabet, we obtain an automaton model, whose final states indicate the recognition or verification of a higher-level concept. The investigation of this will be future work. The point, we would like to emphasize here, is that the inference of automata is a contribution to the idea to learn abstract operational concepts, such as *move through door, turn left, and stop in front of the cupboard* (see [10]), which will be used flexibly, also in unknown environments. So our focus is put on closing the gap between the numerical representation of sensor data and the logic-based description of abstract concepts, which can be understood and used by a human user, in order to guide the robot. This idea distinguishes this work from the numerous approaches in the field of robotics, which, using sensor data, try to localize the robot with the help of statistical methods. Crowley [3], Leonard [8], and Curran[4], for example, use the Kalman filter to estimate the robot position from sensor data in known environments, which are represented by geometric descriptions. The goal of all of these approaches is different from ours in the following sense: They try to improve processing the sensor data at one level, whereas we aim at bridging the gap between low-level representations of sensing and action and the high-level representations of operational concepts. This is done in several steps, by applying different algorithms, which learn, how to derive higher-level concepts from lower-level ones. The approach of inferring automata, presented in this paper, is integrated in this framework. An interesting point for future work would be, to see, whether the approaches, mentioned above, could be, in principle, integrated in our framework, as they have the advantage of considering sensor noise and sensor failure, respectively.

9 Acknowledgements

Part of this work was done during the author's stay at the University of Torino. The author would like to thank A. Giordana, L. Saitta, and all members of the machine learning group for fruitful discussions during the stay at Torino. Thanks also to St. Wessel for providing the algorithm for calculating basic features.

References

[1] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[2] E. Charniak. Passing markers: A theory of contextual influence in language comprehension. *Cognitive Science*, 7, 1983.

[3] J. L. Crowley. World modeling and position estimation for a mobile robot usig ultrasonic ranging. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 674–680, 1989.

[4] A. Curran and K. J. Kyriakopoulos. Sensor-based self-localization for wheeled mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 8–13, 1993.

[5] T. Dean, K. Basye, and L. Kaelbling. Uncertainty in graph-based map learning. In J. H. Connell and S. Mahadevan, editors, *Robot Learning*, chapter 7, pages 171–192. Kluwer Academic Press, 1993.

[6] J. H. Gennari, P. Langely, and D. Fisher. Models of incremental concept formation. *Machine Learning*, 40:11–61, 1989.

[7] J. A. Hendler. Integrating marker-passing and problem solving. In A. Tate J. Allen, J. Hendler, editor, *Readings in Planning*, pages 275–287. Morgan Kaufmann, 1990.

[8] H. F. Durrant-Whyte J. J. Leonard. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7:376–382, 1991.

[9] F. Jelinek. Continuous speech recognition by statistical methods. *Proc. of the IEEE*, 64:532–556, 1976.

[10] V. Klingspor, K. Morik, and A. Rieger. Learning operational concepts from sensor data of a mobile robot. (submitted to Machine Learning Journal), September 1994.

[11] K. Morik and A. Rieger. Learning action-oriented perceptual features for robot navigation. In *Proc. of the 1st European Workshop on Learning Robots*, 1993. also available as Research Report 3, FB Informatik LS 8, Universität Dortmund.

[12] R.E. Schapire R. L. Rivest. Inference of finite automata using homing sequences. In R.L. Rivest S. J. Hanson, W. Remmele, editor, *Machine Learning: From Theory to Applications*, pages 51–73. Springer, 1993.

[13] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[14] Wei-Min Shen. Learning finite automata using local distinguishing experiments. In R. Bajcsy, editor, *Proc. IJCAI 1993*, pages 1088–1093. Morgan Kaufmann, 1993.

[15] E. Sommer. Fender: An approach to theory restructuring. In N. Lavrac and St. Wrobel, editors, *Proc. of the European Conference on Machine Learning (ECML-95)*. Springer Verlag, 1995.

[16] W. Tzeng. Learning probabilistic automata and Markov chains via queries. *Machine Learning*, 8:151–166, 1992.

[17] St. Wessel. Lernen qualitativer Merkmale aus numerischen Robotersensordaten. Master's thesis, Universität Dortmund, 1995. in German.

[18] S. Wrobel. *Concept Formation and Knowledge Revision.* Kluwer Academic Publishers, 1994.