Models for Prediction

P@trik Haslum
Department of Computer Science, Linköping University
pahas@ida.liu.se

Abstract

Prediction is found to be a part of many more complex reasoning problems, *e.g.* state estimation, planning and diagnosis. In spite of this, the prediction problem is rarely studied on its own. Yet, there appears to be a wide range of choices for the design of the central component in a solution to this problem, the predictive model. We examine some of the alternatives and, as a case study, present two different solutions to a specific prediction problem that we have encountered in the WITAS UAV project.

1 Introduction

The capability to predict the future development of a dynamic environment from information about the past and present is an essential component of many reasoning tasks. That it appears as a component in the solution of a more complex problem is natural, since the value of prediction lies in its use to support decisions in *e.g.* planning or diagnosis, but because of this, the problem of prediction in itself is more rarely studied. It appears that there is a wide range of choices in the design of a predictive mechanism, and that in many cases the component that manages prediction in *e.g.* a planning system could be substituted without major changes to the remaining parts of the system. Because of this, it makes sense to investigate the "design space" of predictive mechanisms, as well as classes of environments for which one mechanism or another may be more suitable.

In the remainder of this section, we examine some of the reasoning tasks in which prediction plays a role, while section 2 examines some of the choices that are to be made in the design of a predictive mechanism. In section 3, we present as a case study a prediction problem along with two different solutions, that is, models. The problem, taken from the WITAS project[1], is to predict the movements of a car in road network, and appears in the context of the task of planning a search strategy to locate a specific car.

[1]The WITAS project studies architectures and techniques for intelligent autonomous systems, centered around an unmanned aerial vehicle (UAV) for traffic surveillance. For an overview, see *e.g.* (Doherty *et al.*, 2000) or `http://www.ida.liu.se/ext/witas/`.

1.1 Sensing: State Estimation and Reidentification

State estimation may be described as the problem of constructing a coherent and as correct as possible world view from a sequence of sensor readings disturbed by noise. Here, "world view" may be anything from the position of a mobile robot to classifications and motion vectors of all objects detected in a stream of images.

Predicting the evolution of the world from the time of one reading to the next introduces a dependency between measurements, which allows more information to be used and a more accurate estimate to be made (as long as the predictive model is not more wrong than the sensors). Examples include the Kalman filter (Kalman, 1960), Markov localization methods (see *e.g.* Thrun (2000) for a survey) and their combination (Thiébaux and Lamb, 2000).

Reidentification

A closely related problem is found in the process of anchoring symbolic referents to sensory perceptions (objects): Reidentification of a previously observed object[2].

Consider, for example, tracking an object by vision (a problem that arises also in the WITAS UAV application).When a tracked object is lost from sight and later seen again, its visual appearance may have changed, due to *e.g.* perspective or light conditions changing, so much that comparing visual characteristics like shape and color is not sufficient to reconize the object with a usefull degree of certainty. Again, prediction can introduce a dependency between the past and present observations of the object, increasing certainty in the decision of whether to match the two.

In the work of Huang *et al* (1997; 1999) on tracking cars by means of cameras stationed along a freeway, prediction takes the form of appearance probabilities that

[2]Coradeschi and Saffiotti (2000) describe instead the problem as finding the best match, if any, for a previously anchored referent among a set of perceived objects, and call it *reacquisition*.

"define how an object observed at some point in the past can be expected to appear at some point in the future" and depend both the object and the camera positions (for example, the objects estimated velocity and the distance to the next observation point determine the expected time of reappearance). Coradeschi and Saffiotti (2000) define a framework in abstract, and leave the concrete form of the predictive model open.

1.2 Control: Reaction and Anticipation

Although tracking control, *i.e.* control problems in which the reference point is constantly changing, is inherently a reactive activity, predicting the near future reference values can improve the precision and economy of the controller.

Taking a problem from the WITAS project, consider the UAV tracking a car traveling on a road: It is desirable to keep the UAV at some angle, say 25° - 45°, behind the tracked car *w.r.t.* the cars direction of movement. Matching the expected future velocity of the car instead of that currently measured reduces the need for abrupt course corrections (reducing the risk of losing track of the car due to sudden camera movements) and the risk of overshoot. Even a very simple predictive model (essentially assuming that the car maintains its current x, y velocity) allows the UAV to approach the desired position and velocity much more smoothly. A more elaborate model could take into account the shape of the road and behaviour such as slowing down when approaching an intersection.

The principle applies on a larger time scale as well: If events in the near future can be predicted, more time is available to prepare a reaction and the actions taken can be better tailored to the developing situation. An example is Propice-Plan (Despouys and Ingrand, 1999; Despouys, 2000), a task execution and monitoring system. The system applies short-term prediction to make a more informed choice of which of several alternative procedures to apply to achieve a given goal, as well as to anticipate, and if possible avoid, failures before they become a fact.

1.3 Planning: Verification and Guidance

In most cases the aim of planning is to produce correct plans, *i.e.* plans that are if not certain at least highly likely to achieve the intended goals. Verifying a plan amounts to deciding, given facts of the initial situation, whether every possible (or at least the most likely) execution path of the plan does indeed lead to the achievement of the intended goals, and this is a prediction problem.

For the STRIPS class of planning domains the problem is extremely simple, since information is assumed complete and accurate and the dynamics are deterministic, and therefore very rarely explicitly considered[3]. For planning in domains in which facts and outcomes of actions are unknown or uncertain, verification naturally becomes harder, as can be seen in that it contributes a significant part to the complexity of planning algorithms for such domains. Contrast, for example, solution extraction procedures in GRAPHPLAN (Blum and Furst, 1997) and the conformant planner CGP (Smith and Weld, 1998). In Markov decision problems (Puterman, 1994; Kaelbling *et al.*, 1998), which are frequently used for planning under uncertainty, the aim is to find a policy that maximizes the expected utility or reward. Taking expectations implies the existence of a (probabilistic) predictive model.

Certain "repair based" planners, *e.g.* XFRM (Beetz and McDermott, 1992) and WEAVER (Blythe, 1998), make explicit use of prediction to detect flaws in a candidate plans and guide the selection of repairs. The prediction procedure in XFRM (McDermott, 1994) is based on a description by probabilistic rules and events appearing randomly with a given intensity[4]. WEAVER interleaves contingent planning in a simplified domain and predicting likely failures in the resulting plan. Repair consists in "de-simplifying" relevant parts of the domain and replanning.

1.4 Execution Monitoring and Diagnosis

Execution monitoring and diagnosis is often cast as the problem of detecting faulty behaviour in a system and isolating the malfunctioning system component responsible for it.The commonly adopted definition (*e.g.* Dean and Wellman (1991)) of "faulty behaviour" is behaviour that deviates from the expected behaviour[5].

In the presence of fault models, the diagnosis task closely resembles discrete state identification, *i.e.* determining which of a set of different state trajectories or operating modes of the system best explain (or sometimes, are consistent with) the given observations, based on models that predict system behaviour in each mode. It is also in this area that perhaps the widest range of different kinds of predictive models have been considered. Examples include probabilistic and hybrid transition systems (DeCoste, 1990; Williams and Nayak, 1996; McIlraith *et al.*, 2000), logical action theories (McIlraith, 1998) and specifications in fuzzy linear temporal logic (Ben Lamine and Kabanza, 2000)[6].

2 The Predictive Model

Without a model there can be no prediction. The model contains the knowledge, or assumptions, about the dy-

[3] Though note that if the steps of the plan are only partially ordered, determining what holds and what does not at a particular point in execution of the plan is not as easy (Chapman, 1987).

[4] Although we have not formally proven it, the prediction model used by XFRM appears to be a Markov jump process (see section 3.5).

[5] Saffiotti (1998) takes the, in a sense more general, view that monitoring consists not in deciding if the observed situation is the expected, but whether the executing plan or program remains relevant in this situation.

[6] The logic (LTL) and progression algorithm used by Ben Lamine and Kabanza (2000) is similar to the logic MITL used in our expectation-based model (see section 3.2).

namics of the environment that allows the prediction mechanism to conclude something more than that "anything can happen". To be more precise, a predictive model describes what can *possibly* happen, and optionally what is *likely* to happen.

This section examines some issues in the construction, representation and use of models for prediction. The representation of the prediction model constrains the entire predictive mechanism, *e.g.* how the model can be acquired, what results can be obtained from it and how those results are computed. Naturally, the model representation may equally be said to be determined by the results desired, the acquisition methods available, *etc.* Nevertheless, representation will be the main starting point of the discussion, and limited to models based on the notion of *state*[7].

2.1 Representations of State and Time

What aspects are to be included in a state and what form they take, *e.g.* whether measurements should be treated as continuous or discrete, is largely determined by the constraints and needs of the application.

Regarding time, not so many different views can be adopted: (*1*) Time may be viewed as a continuum, in which the state evolves continuously. The typical example of a model of this kind is a differential equation (though note that computation with such models usually is based on discrete approximations). (*2*) Time may be viewed as a continuum, in which the state remains steady until it changes instantaneously. Examples of such models include timed and hybrid transition systems (Alur and Dill, 1994; Alur *et al.*, 2000), partially ordered plans and formalisms like the event calculus (Kowalski and Sergot, 1986). Constraints of some form appear to be the most common method of representation and reasoning with time in this kind of model. (*3*) Time may be view as a sequence of discrete "points", with a state at each point and state changes between points. This is the most common kind of model, examples including classical plans, diagnosis models, and more.

When discrete time is viewed as an abstraction of continuous change, which is often the case, an implicit assumption is that the discretization is fine enough, *i.e.* that the changes between one point and the next amount to no more than the change from the state at the first point to the state at the next.

2.2 Representations of Uncertainty

In all but the most trivial cases, predicting the future is an uncertain business. At least three approaches to dealing with this uncertainty have been used: (*1*) Ignore it. In, for example, planning it is of course the case that any part of a plan always may fail (and fail in a number of different ways), but if failures are very rare, it may not be worthwhile to plan for the contingencies that failures cause and hence to view the expected outcome of every action as certain. (*2*) Treat it as non-determinism, that is, to consider *all* possible outcomes regardless of likelihood. This view is necessary when absolute certainty is required, *e.g.* in formal verification (Clarke *et al.*, 1999), and is also adopted in conditional planning (Peot and Smith, 1992; Collins and Pryor, 1995; Weld *et al.*, 1998). (*3*) Quantify it. This approach is at its best when reliable measures or estimates, such as statistics, of any uncertainties are available, and when "sufficient" or "best possible", rather than absolute, certainty is required of the prediction. Examples are found in most of the applications in the previous section, *e.g.* Kalman filters in state estimation and MDPs in planning. The most common way to measure uncertainty is by probabilities, though alternatives, *e.g.* fuzzy logic, exist.

From an algorithmic point of view, the difference between non-deterministic and probabilistic representations seems to be small. For example, dynamic programming methods for solving MDPs translate into algorithms for non-deterministic planning by considering at each step the worst outcome rather than each outcome weighted according to probability (Koenig and Simmons, 1995; Bonet and Geffner, 2000). The difference lies in the interpretation of the result. An example is the "strong cyclic plan" of Cimatti *et al.* (1998), which is guaranteed to reach its goal assuming no execution path of the plan is ignored forever. In other words, the plan works if actions are random (with non-zero probability of each outcome) but may loop forever if actions are non-deterministic.

2.3 Model Acquisition

An important aspect of the construction of a predictive model is its source: It may be computed, or learned, from data, it may be an encoding of "expert knowledge", or it may have been derived from a model of another kind. Representations are often more or less suited to each mode of acquisition. Probabilities, for example, can be accurately estimated from data by statistical methods, while people, experts or not, are notoriously poor at estimating probabilities (Tversky and Kahneman, 1974).

2.4 Computation with the Model

The purpose of the predictive model is to *compute* predictions. However, the importance of computational properties such as scaling, numerical stability or even decidability depends on the application, specifically on what questions the prediction is meant to answer.

For example, contingent planning is in most cases restricted to relatively simple forms of plans, typically trees of finite depth, because every additional complication in a plan makes the problem of verifying it, for every possible outcome, immediately much harder. The XFRM planner, on the other hand, never considers every

[7] From systems theory, we know that for any general system a representation by state object and response function can be found (Mesarovic and Takahara, 1975), although Rosen (1991) argues that certain systems, in particular living beings, can not be represented in such a "mechanical" way.

outcome but only a random sample, and therefore can allow much more complex plan forms.

Another example is the model representation developed in sections 3.2 – 3.3, which has its roots in the area of formal verification but which relaxes several restrictions. The reason is that those restrictions are necessary to make verification problems, which concern the infinite horizon behaviour of the modeled system, decidable, while the application we study is concerned only with finite predictions.

3 Case Study: Predicting Car Movement

As a case study, we take a prediction problem from the WITAS project and develop two different solutions, *i.e.* predictive models. Because we currently lack the data necessary to build accurate predictive models for the domain, and the possibility to experimentally evaluate them, models are only sketched. The focus is on the representational choices and the consequences of those choices. Strategies for evaluating model designs are discussed in section 4.

A typical task for the traffic monitoring UAV may be to find a specific car that has previously been seen. This requires planning a search strategy. Assuming that the car is behaving independently of the UAV, *i.e.* that it is neither cooperative nor adversarial, the problem decomposes in two parts: (*1*) Determine locations and times where the car is most likely to be intercepted[8]. (*2*) Plan a flight path to cover as many locations/times as possible. Here, we focus on the first part.

The problem, thus, is to predict the most likely present locations of a car, from one or more observations of its position some time in the past[9]. A detailed map of the road network, with some "static" information about car behaviour, *e.g.* average speeds and traffic volumes, is assumed to be available.

Both models are based on a discrete state representation, where the position of the car is described only by what road or intersection it is in (the exact position along a road will quickly become too uncertain to be of any use). Because of the inherent uncertainty and large differences in the time the car stays in different states, a continuous, event based, representation of time is adopted in both models. The main difference is in the way they represent uncertainty.

3.1 Case I: Expectations as Constraints

In the first model, world dynamics are represented by a timed transition system. The dynamics of the world determine only what developments, that is sequences of states and events, are possible. In addition, we *expect* the world state to develop along certain lines, for instance that cars do not suddenly stop or make U-turns (or at least not more than one in quick succession). A development is *normal* to the degree that it satisfies our expectations. These expectations are constraints on state/event sequences, expressed in a temporal logic.

The next section introduces the necessary technical background, while the model is described in the following section. Discussion of the advantages and disadvantages of the representation is in section 4.

[8] Lacking any information about the likely present position, the best strategy is "submarine search", *i.e.* searching in a pattern spiraling outwards from the last known position.

[9] The same problem but on a much smaller time scale is considered by Forbes *et al.* (1995). Their solution uses probabilistic networks to represent a discrete-time model.

3.2 Technical Background

This section introduces timed transition systems and Metric Interval Temporal Logic (MITL). Details can be found in *e.g.* Alur (1999) and Emerson (1990).

Timed Transition Systems

Timed transition systems (Alur and Dill, 1994) are essentially finite state automata, augmented with time constraints of two kinds: A transition can have a time window in which it is possible to make the transition and a state can have a maximal time that the system may remain in the state before it has to exit by some transition.

Let $\mathbb{R}^+$ denote real numbers $\geqslant 0$, with a special symbol ∞ for infinity. Formally, a timed transition system $S = (Q, R, C, L)$, consists of a set of *states*, Q, and a *transition relation*,

$$R \subseteq \Sigma \times Q \times Q \times \mathbb{R}^+ \times \mathbb{R}^+$$

where Σ is some set of *event* labels. The interpretation is that if $(a, q, q', t, t') \in R$, the system may transit from state q to q' in response to the event a in the time interval $[t, t']$, relative to the time that the system entered q. Time constraints of the second kind are specified by the function $C : Q \longrightarrow \mathbb{R}^+$. States are labeled with *properties* from a set P via a function $L : Q \longrightarrow 2^P$ (often the set of states is 2^P, *i.e.* a state is defined by its properties).

Like a finite automaton accepts a set of strings over its alphabet, a timed transition system "accepts" a set of histories: A *development* is a sequence of states and events marking state transitions, $d = q_0, a_0, q_1, a_1, \ldots$, with an associated function $T : d \longrightarrow \mathbb{R}^+$ that tells the starting time of each state, such that

(*i*) for $i \geqslant 0$, there exists $t, t' \in \mathbb{R}^+$ such that $R(a_i, q_i, q_{i+1}, t, t')$ and $T(q_i)+t \leqslant T(q_{i+1}) \leqslant T(q_i)+t'$, and

(*ii*) for $i \geqslant 0$, $T(q_{i+1}) \leqslant T(q_i) + C(q_i)$[10].

[10] Two additional properties are usually required of a timed transition system: *Executability*, which is the requirement that any finite prefix satisfying conditions (*i*) and (*ii*) can be extended to an infinite development, and *non-zenoness*, which is the requirement that the system does not make an infinite number of transitions in finite time. Because we shall only be concerned with finite (in time and number of events) developments, these properties are not important to us.

The duration of a state q_i is denoted $D(q_i) = T(q_{i+1}) - T(q_i)$.

Even when only finite development prefixes starting in a specific state q_0 are considered, the set of possible developments is uncountable, since the starting time of any state in a development can change by an arbitrarily small amount. To be able to enumerate finite developments, we have to adopt a more compact representation: A set of developments that differ only on state starting times are represented by the sequence of states and events, $d = q_0, a_0, \ldots, q_n$, and a set of constraints on the starting times $T(q_0), \ldots, T(q_n)$. The time constraints are managed in a temporal constraint network (TCN) (Dechter *et al.*, 1991).

Metric Interval Temporal Logic

The language of MITL (Alur *et al.*, 1996) consists of atoms P, *i.e.* the properties of states, propositional connectives and the temporal operators *always* ($\Box_{[t,t']}\varphi$), *eventually* ($\Diamond_{[t,t']}\varphi$), *next* ($\bigcirc_{[t,t']}\varphi$) and *until* ($\varphi \, \mathcal{U}_{[t,t']} \, \psi$). The intervals adjoined to the operators take values in $\mathbb{R}^+$ and express metric temporal restrictions[11].

Formulas in MITL are evaluated over a time development (d, T) as follows: Let d^i denote the suffix of d starting with the ith state. A formula φ not containing any temporal operator holds in the development d^i iff φ holds in the state q_i. The conditions for temporal formulas are

- $\Box_{[t,t']}\varphi$ holds in d^i iff φ holds in every d^k such that $T(q_i)+t \leqslant T(q_k) \leqslant T(q_i)+t'$ or such that $T(q_i)+t < T(q_{k+1}) \leqslant T(q_i) + t'$.
- $\Diamond_{[t,t']}\varphi$ holds in d^i iff there exists an q_k such that $T(q_i) + t \leqslant T(q_k) \leqslant T(q_i) + t'$ or $T(q_k) < t < T(q_{k+1})$, and such that φ holds in d^k.
- $\bigcirc_{[t,t']}\varphi$ holds in d^i if φ holds in q_{i+1} and $T(q_i)+t \leqslant T(q_{i+1}) \leqslant T(q_i) + t'$.
- $\varphi \, \mathcal{U}_{[t,t']} \, \psi$ holds in d^i iff there exists a q_k such that $T(q_i) + t \leqslant T(q_k) \leqslant T(q_i) + t'$ or $T(q_k) < t < T(q_{k+1})$, ψ holds in d^k and φ holds for all d^j with $i \leqslant j < k$. begins to hold.

Connectives are interpreted as in ordinary logic.

Progression of MITL Formulas

The above conditions for the truth of an MITL formula are written for infinite developments. For a finite (prefix) development, we say a formula holds iff it holds in some continuation of the development, according to the above conditions. To determine if this is the case, the formula progression algorithm of Bacchus and Kabanza (1996) can be used.

The algorithm takes an MITL formula φ and a state q with duration $D(q)$ and returns an MITL formula φ' such that φ' holds in the next state iff φ holds in q. If the input formula is not true in any continued development, the result of progression is equivalent to FALSE.

The basic progression algorithm assumes that $D(q)$ is known exactly, but as explained at the end of the previous section we have to represent sets of developments by a combination of a state/event sequence and constraints on state starting times. Therefore, the algorithm has to be extended to take as input a set of time constraints, C, and return the set of *all* solutions, $\{(\varphi'_1, C'_1), \ldots, (\varphi'_k, C'_k)\}$, where each C'_i is a set of additional time constraints consistent with C and φ'_i is the result of progressing the input formula φ under constraints $C \cup C'_i$.

Because the progression algorithm works recursively by cases, the extension is straightforward (though somewhat complicated in practice): For example, progressing the formula $\varphi = \Box_{[5,9]} \bigcirc_{[0,4]} p$ through state q results in

(a) $\Box_{[5-D(q),9-D(q)]} \bigcirc_{[0,4]} p$, if $D(q) < 5$;

(b) $\psi' \wedge \Box_{[0,9-D(q)]} \bigcirc_{[0,4]} p$, if $5 \leqslant D(q) < 9$;

(c) ψ', if $9 \leqslant D(q)$

where ψ' is the result of progressing $\bigcirc_{[0,4]}p$ through q: (d) p iff $D(q) \leqslant 4$, and (e) FALSE if not. If the input set of constraints is $C = \{0 < D(q) < 7\}$, the extended algorithm returns two solutions:

$\varphi'_1 = \Box_{[5-D(q),9-D(q)]} \bigcirc_{[0,4]} p$ with the associated constraint set $C'_1 = \{D(q) < 5\}$ (case (a)), and

$\varphi'_2 = \text{FALSE} \wedge \Box_{[0,9-D(q)]}\alpha$ along with the constraint set $C'_2 = \{4 < 5 \leqslant D(q)\}$ (cases (b) and (e)).

The combination of cases (b) and (d) is inconsistent (since it requires $5 \leqslant D(q) \leqslant 4$), and case ($c$) contradicts the input constraints.

In general, an MITL formula defines a tree-like structure of possible progressions with associated time constraints, and the extended algorithm retrieves all consistent paths through this tree.

3.3 The Model

We sketch a predictive model based on the transition/expectation representation in three steps: First, the properties and dynamics of states and second, the constraints on developments that represent expectations. Last is a brief discussion of how predictions are computed using the model and how predictions are used to answer queries.

State Properties and Dynamics

Properties and dynamics of states are represented by a timed transition system. Let the set of atoms comprise $\mathrm{in}(\mathrm{r}_i)$, $\mathrm{in}(\mathrm{x}_i)$ for all roads r_i and intersections x_i in the map, denoting the location of the car, $\mathtt{at\_start}(\mathrm{r}_i)$ and $\mathtt{at\_end}(\mathrm{r}_i)$, indicating the car is at the very beginning or end of road r_i and an atom $\mathtt{moving}$. Examples of transitions are

$$
\begin{aligned}
&(\mathtt{enter}(r,x),\\
&\quad \mathtt{in}(x) \wedge \mathtt{moving} \wedge \mathtt{starts}(r,x),\\
&\quad \neg\mathtt{in}(x) \wedge \mathtt{in}(r) \wedge \mathtt{at\_start}(r),\\
&\quad 1, \infty) \in R
\end{aligned}
$$

[11] The standard definition of MITL disallows singleton intervals, *i.e.* of the form $[t,t]$, to make certain questions regarding the infinite horizon behaviour of the system decidable. Again because we shall be dealing only with finite developments, we ignore this restriction.

$$(\texttt{drive}(r),\; \texttt{in}(r) \wedge \texttt{at\_start}(r) \wedge \texttt{moving},\; \neg\texttt{at\_start}(r) \wedge \texttt{at\_end}(r),\; t^{min}(r), \infty) \in R$$

$$(\texttt{stop}, \texttt{moving}, \neg\texttt{moving}, 0, \infty) \in R$$

for all roads r and intersections x in the map[12]. Because the drive transition represents the car traveling the entire length of the road the lower bound can be made a function of the road, based on *e.g.* length, shape and volume of traffic. Matching upper bounds are provided by state constraints:

$$C(\texttt{in}(r) \wedge \texttt{moving}) = t^{max}(r)$$

$$C(\texttt{in}(x) \wedge \texttt{moving}) = 5$$

expressing that the car can only remain in the same location for a certain time and still be considered moving.

The Expectation Hierarchy

Next, the model is enriched with constraints representing expectations on developments, constraints expressed as MITL formulas. For example, the expectation that the car does not suddenly stop or make a U-turn can be formulated

$$\Box_{[0,\infty]}(\texttt{in}(r) \wedge \texttt{moving} \wedge \texttt{ends}(r,x) \rightarrow (\texttt{in}(r) \wedge \texttt{moving}\,\mathcal{U}_{[0,t^{max}_{norm}(r)]}\,\texttt{in}(x))) \quad (1)$$

stating that "it's always the case that if the car is in road r and moving, it remains in the road and moving until it's in the intersection x at the end of r". Furthermore, $t^{max}_{norm}(r)$ sets an upper bound on the time that the car is expected to remain in the road. To express a lower bound, a more complicated construction is needed:

$$\Box_{[0,\infty]}((\texttt{in}(x) \wedge \texttt{starts}(r,x) \wedge \bigcirc_{[0,\infty]}\texttt{in}(r)) \rightarrow \bigcirc_{[0,\infty]}\Box_{[0,t^{min}_{norm}(r)]}\texttt{in}(r)) \quad (2)$$

This states that "(it's always the case that) if the car is in intersection x and in the next state in road r beginning at x, it is in r at all times in the interval $[0, t^{min}_{norm}(r)]$ from the beginning time of the next state".

We expect of course that where the car goes depends on where it is going:

$$\Box_{[0,\infty]}((\texttt{in}(r) \wedge \Diamond_{[0,\infty]}\texttt{in}(r') \wedge \texttt{distance}(r,r'') < \texttt{distance}(r',r'')) \rightarrow \neg\texttt{destination}(r'')) \quad (3)$$

This states that if the car is first in road r and later in road r', and if the distance from r' to r'' is greater than the distance from r to r'', then the cars destination is not in road r''. This is a strong expectation, not allowing the car any detours, but weaker forms can also be expressed. In difference to properties like location, the atom destination(r) can not be observed, so expectation (3) may seem useless, but since we assume that the destination does not change over time, if a sequence of observations is available, the expectation can be used to rule out some possibilities, in effect inferring what the potential destinations are.

As mentioned, a development is normal to the degree that it satisfies expectations, but since satisfaction of MITL formulas is strictly Boolean, "degree" can only mean "number of". At the same time, we have more confidence in some expectations than in others. The solution to both problems is to arrange expectations in a hierarchy of some sort, the simplest being a sequence $\varphi_1, \ldots, \varphi_n$ of decreasing confidence. A development satisfying expectations $\varphi_1, \ldots, \varphi_k$ but not φ_{k+1} is then said to be *normal at level k*.

For example, a weaker form of expectation (1) not specifying an upper time bound may be ordered before (1) and (2) and after them (3), since that is a very strong constraint. In general, ordering expectations according to confidence is a very important, and often difficult, issue in building a model of this kind.

Computing Predictions

How is the answer to the question "where is the car expected to be now?" computed from a set of past observations and a model as described above? The answer amounts to finding the position of the car in all states that may exist at time "now" in the set of most normal developments starting from the state at the last observation. That is, it is a reachability problem and can be solved by search in the tree of finite developments.

If a sequence of observations is available, progressing expectations like (3) through the observed states leads to constraints on the possible values of unobservable properties, such as destination. The inferred knowledge can then be used to limit the prediction search.

The tree of developments can grow exponentially with the depth, *i.e.* prediction time span. However, if uncertainty is low, for example due to strong expectations, the branching factor comes close to 1, resulting in linear growth.

3.4 Case II: A Markov Process

The second model is a Markov process, *i.e.* a transition system that has probabilities associated with the transitions out of each state. Because much of the uncertainty in predicting the cars movement is related to time, the time in each state in a development is a continuous probability distribution instead of a simple discretization. The state duration distribution is, for computational reasons, limited to the exponential, and since this distribution is unsuited for describing the domain a certain "encoding trick" must be used.

Again, the next section introduces technical background, while the model and its use for prediction is described in the following one. Discussion is in section 4.

[12] For compactness and modularity, the states involved in each transition are only partially described and a STRIPS style (minimal change) semantics is assumed. This necessitates an additional complication: The time window of a transition is not relative to the time of the latest event, but to the time at which the transition became enabled. Besides state atoms, conditions also include some "static" facts, such as starts(r,x) and ends(r,x), describing the road network.

3.5 Technical Background

This section briefly introduces the continuous-time Markov jump process, some simple methods for parameter estimation and the phase method. For a more detailed treatment, see *e.g.* Tijms (1994).

Markov Chains

A Markov chain is a discrete time random process. Formally it consists of a finite or countably infinite set of *states* $\mathcal{X}$ and a sequence of random variables $X_0, X_1, \ldots$, taking values in $\mathcal{X}$, representing the state of the process at each stage. The process being Markov, it satisfies

$$\mathbf{P}(X_i = x_i \mid X_0 = x_0, \ldots, X_{i-1} = x_{i-1}) = \mathbf{P}(X_i = x_i \mid X_{i-1} = x_{i-1}) \quad (4)$$

i.e. the probability of state x materializing at stage i depends only on the state of the process at stage $i-1$. The $\mathbf{P}(X_i = x_i \mid X_{i-1} = x_{i-1})$ are called the *transition probabilities*, and if they are identical for all i the process is *stationary*. In this case, the abbreviations

$$P(x, x') = \mathbf{P}(X_{i+1} = x' \mid X_i = x)$$
$$P^n(x, x') = \mathbf{P}(X_{i+n} = x' \mid X_i = x)$$

are used.

Let $\pi_i(x) = \mathbf{P}(X_i = x)$, $x \in \mathcal{X}$, denote the distribution of the process at stage i. Because of stationarity and the Markov property, π_i and the transition probabilities determine the distribution at every subsequent stage:

$$\pi_{i+k}(x) = \sum_{x' \in \mathcal{X}} \pi_i(x') P^k(x', x) \quad (5)$$

A stationary Markov chain is therefore characterized by the transition probabilities and the initial distribution π_0.

The Markov Jump Process

For the continuous time case, let $X(t)$, $t \geqslant 0$ denote the state of the process at *time* t. The discrete sequence obtained by "sampling" the state of the process just after every state change is called the *embedded chain* of the process.

A *Markov jump process* is a random process $X(t), t \geqslant 0$ such that the embedded chain is a stationary Markov chain and such that the time in each stage, T_i, satisfies the conditions:

(*i*) T_i is exponentially distributed with a mean $\frac{1}{\lambda(x)}$ that depends only on the state of the process at the stage, *i.e.* $T_i = T(X_i)$ ($\lambda(x)$ is called the *leaving rate* of state x).

(*ii*) The number of stages that occur in a finite time interval is finite with probability 1.

If the leaving rate is $\lambda(x) = \lambda$ for all $x \in \mathcal{X}$, the process is said to be *uniform*. By convention, the transition probability $P(x, x) = 0$ for all $x \in \mathcal{X}$ to ensure that the time $T(x)$ is unambiguously defined.

The leaving rates and the transition probabilities do not in the general case uniquely determine a continuous-time Markov jump process. However, when the state space is finite, the process is unique and a distribution $\pi_t(x) = \mathbf{P}(X(t) = x)$, $x \in \mathcal{X}$ determines the distribution for all $t' > t$. If the process is also uniform, with leaving rate λ, this distribution is

$$\pi_{t+\Delta}(x) = \sum_{x' \in \mathcal{X}} \pi_t(x') \sum_{k \geqslant 0} F(k, \lambda\Delta) P^k(x', x) \quad (6)$$

where $F(k, \mu)$ is the Poisson probability function with mean μ[13].

3.6 The Phase Method

The main limitation of the Markov jump process, from the point of view of our application, is the restriction to exponentially distributed stage time. To circumvent it an "encoding trick", invented by A. K. Erlang in the early 20th century and known as the phase method, is needed.

Let $F(x)$ be any non-negative distribution and for any fixed $\delta > 0$ let

$$F_\delta(x) = \sum_{k=0}^{\infty} p_\delta(k) \left(1 - \sum_{i=0}^{k-1} e^{-\frac{1}{\delta}x} \frac{(\frac{1}{\delta}x)^k}{k!} \right) \quad (7)$$

where $p_\delta(k) = F(k\delta) - F((k-1)\delta)$ for $k = 1, 2, \ldots$. Then $\lim_{\delta \to 0} F_\delta(x) = F(x)$, *i.e.* any non-negative distribution can be approximated with arbitrary precision by a sum of exponentially distributed variables all with the same rate parameter, $\frac{1}{\delta}$.

If the stage time $T(x)$ of a continuous-time Markov process is, for instance, normally distributed with mean $\mu(x)$ and standard deviation $\sigma(x)$, both of which depend on the state[14], it may instead be seen as a sum of k exponentially distributed "phases" with mean $\frac{1}{\delta}$, with probability

$$p_\delta(k) = \Phi(\frac{k\delta - \mu(x)}{\sigma(x)}) - \Phi(\frac{(k-1)\delta - \mu(x)}{\sigma(x)}) \quad (8)$$

for each k, where $\Phi(x)$ denotes the standard normal distribution. Thereby, the process can be approximated by one that is a uniform Markov jump process.

Estimating Processes Parameters

It is a simple generalization to let the transition probabilities depend on an additional, unknown but time constant, parameter, Y, taking values in some finite, discrete domain $\mathcal{Y}$:

$$P_y(x, x') = \mathbf{P}(X_{i+1} = x' \mid X_i = x, Y = y) \quad (9)$$

The *a priori* probabilities $\mathbf{P}(Y = y)$, $y \in \mathcal{Y}$ are assumed known. Estimating the likelihood of $Y = y$ after observation of a sequence of states is a straightforward application of Bayes' rule.

---

[13] $F(k, \lambda\Delta)$ may be interpreted as the probability of seeing k events in Δ units of time, if the time between any two events is exponentially distributed with intensity λ, *i.e.* events happen "on average" once every $\frac{1}{\lambda}$ time unit.

[14] Although the normal distribution assigns positive probability to negative values, let us assume $\mu(x)$ and $\sigma(x)$ are such that this probability is negligible.

The distribution of the stage time $T(x)$ can also be made to depend on unknown parameters, in addition to the state. There are many models to chose from, for example $T(x) = F_0 + F_1(x)$ where F_0 and F_1 are independent, normally distributed and the parameters of F_0 the unknown, or $T(x) = F_0 \cdot F_1(x)$ with the same assumptions. The methods available to estimate the unknown parameters vary with the choice of model, but most depend on having observations of $T(x)$ which is possible only if the process is observed continuously so that the time of each state change can be recorded.

3.7 The Model

The state of the process is the road that the car currently occupies (we ignore the intersections). The cars traveling time in road r is a random variable with distribution $T(r)$, containing some unknown parameters. The parameter to the transition probabilities is the cars destination, d.

Because of the use of phases to encode $T(r)$, the state space of the process is the set of pairs (r, k), where r is a road in the map and k the phase counter. The transition probabilities are

$$P_d((r,k),(r,k-1)) = 1 \tag{10}$$

$$P_d((r,0),(r',k)) = p_\delta(r,k)\frac{\frac{1}{\mathtt{distance}(r',d)}}{\sum_{r''}\frac{1}{\mathtt{distance}(r'',d)}} \tag{11}$$

$$P_d((d,0),(d,0)) = 1 \tag{12}$$

where $r' \neq d$ and r'' range over all roads starting from the intersection at the end of r and $k = 1, \ldots$. The expression $p_\delta(r,k)$ denotes the probability of the car staying k phases in road r, given by equation (8).

The process as defined by (10) - (12) has an infinite state space, since the number of phases is, theoretically, unbounded. In practice, however, the probability $p_\delta(r,k)$ is negligibly close to zero for all but a finite range of values of k.

Computing Predictions

Given a model as described above and a set of past observations, to answer the question "Where is the car most likely to be now?" amounts to computing the distribution $\pi(x)$ at time "now", starting from a distribution concentrated to a single state (or a set of states sharing the same road component) at the time of the last observation. This is straightforward, using equation (6).

If a sequence of state changes has been observed, estimates of the values of unknown parameters can be improved. If not, the *a priori* parameter probabilities are used.

4 Evaluation

Searching for criteria for evaluating the two kinds of predictive models described above, it is tempting to suggest "accuracy". This would be a mistake since the accuracy of predictions is a product of the model, not of the way it is represented. In short, it is certainly possible to construct a bad model using even the best representation.

Instead, we suggest looking at three points: (*1*) *Representation*, meaning both adequacy and efficacy. That is, is the representation rich enough to express the required model? Can a model of this kind be built, and maintained, with the available sources? (*2*) *Computation.* Can a prediction algorithm based on this kind of model be made efficient enough? (*3*) *Integration.* Does a model of this kind provide the kind of predictions that the application needs? This is, arguably, the most important point, since the value of prediction lies in its use in a larger context.

4.1 Representation

Regarding point (*1*) we do, as stated at the beginning of this study, not yet have enough data to construct realistic models[15] nor the possibility to test such models against reality. Both the presented models are sketches and make many simplistic assumptions, *e.g.* that drivers navigate rationally. It can be conjectured that statistics, for example on the volume of traffic in different parts of the network, are easier to make use of in the Markov kind of model. A criticism against models of both kinds may be that they do not represent the drivers actions in a natural way, since they both treat events as instantaneous.

4.2 Computation

With regards to point (*2*), there may seem to be no difference, since both methods of computing predictions are clearly exponential. They are, however, exponential in different ways.

Computing predictions, *i.e.* future distributions, in the Markov model is exponential in the dimension of the state space (which, due to the encoding of stage time distributions as phases, also increases with the "span", or difference, over those distributions), but polynomial in the prediction time span and almost independent of the amount of uncertainty in the initial distribution or the transition probabilities. In contrast, the time to search the tree of normal developments in the expectation based model depends most of all on the size of the tree, which is exponential in the "amount of uncertainty" (*i.e.* the branching factor) and the prediction time span, but is on the other hand only linearly affected by the dimension of the state space, *i.e.* the number of state properties.

4.3 Integration

Examining finally point (*3*), the models are roughly equivalent even though they provide different kinds of answers. The Markov model gives a more fine-grained measure of the likelihood of finding the car in any particular location, but this measure is only useful in so far

[15] Models of traffic typically deal only with "collective" behaviour, *e.g.* origin/destination frequencies. Matstoms *et al.* (1996) present the average speed of vehicles as a function of traffic density, so called *volume/delay* functions, for 70 types of roads in Sweden, but without statistics like variance.

as it is reliable, *i.e.* that the model is exact. Else it represents merely a "false accuracy".

4.4 Empirical Evaluation

Prediction mechanisms based on models of both kinds presented here have been implemented and tested on a map of the Revinge test flight area (see Doherty *et al* (2000) for a description). However, this experiment can not really be said to constitute an empirical evaluation, for several reasons: First and foremost, our current models are not realistic. Neither is the test flight area a realistic testing ground, since it is (for safety reasons) closed off and thus does not contain any real traffic, and since it is too small for prediction to have a noticeable effect. For experiments in general, the WITAS project relies to a large extent on realistic simulations, but to realistically simulate traffic we would need to already have a model of driver behaviour. Second, the prediction mechnisms are not integrated in the UAV system, so their impact on the overall system performance can not be assesed.

5 Conclusions

We have tried to show that prediction is frequent as a component in more complex reasoning tasks and that there are in the design of any predictive model many intricate choices, leading to several alternatives. This is our working hypothesis. As an example in favour of this hypothesis, we presented two alternative solutions to the problem of predicting car movement in a road network.

Besides developing and integrating and testing the two presented prediction mechanisms, we seek to test the hypothesis in more cases. Since several examples of problems involving prediction were found in the context of the WITAS UAV project a natural approach is to examine also those prediction problems, looking at what kinds of models are being used and if there are any viable alternatives.

Acknowledgments

Many thanks go to Patrick Doherty and Marcus Bjäreland for their valuable comments on drafts of this paper. Thanks also to the reviewers for pointers to interesting related work. Many ideas presented here, not only the problem treated in the case study, have grown out of experiences gained in the interesting environment created by the WITAS project, to which all members of the project team contribute.

The WITAS project and this research is supported by the *Wallenberg Foundation* and the *ECSEL/ENSYM* graduate studies program.

References

(Alur and Dill, 1994) R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183 – 236, 1994.

(Alur *et al.*, 1996) R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116 – 146, 1996.

(Alur *et al.*, 2000) R. Alur, T.A. Henzinger, and P.S. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181 – 201, 2000.

(Alur, 1999) R. Alur. Timed automata. In *NATO-ASI Summer School on Verification of Digital and Hybrid Systems*, 1999. Available at `http://www.cis.upenn.edu/~alur/Nato97.ps.gz`.

(Bacchus and Kabanza, 1996) F. Bacchus and F. Kabanza. Planning for temporally extended goals. In *Proc. 13th National Conference on Artificial Intelligence (AAAI'96)*, 1996.

(Beetz and McDermott, 1992) M. Beetz and D. McDermott. Declarative goals in reactive plans. In *Proc. 1st International Conference on AI Planning Systems*, 1992.

(Ben Lamine and Kabanza, 2000) K. Ben Lamine and F. Kabanza. History checking of temporal fuzzy logic formulas for monitoring behavior-based mobile robots. In *Proc. IEEE International Conference on Tools with Artificial Intelligence*, 2000.

(Blum and Furst, 1997) A.L. Blum and M.L. Furst. Fast planning through graph analysis. *Artificial Intelligence*, 90(1–2):281 – 300, 1997.

(Blythe, 1998) J. Blythe. *Planning under Uncertainty in Dynamic Domains*. PhD thesis, Carnegie Mellon University, 1998.

(Bonet and Geffner, 2000) B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. 5th International Conference on Artificial Intelligence Planning and Scheduling*, 2000.

(Chapman, 1987) D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333 – 377, 1987.

(Cimatti *et al.*, 1998) A. Cimatti, M. Roveri, and P. Traverso. Automatic OBDD-based generation of universal plans in non-deterministic domains. In *Proc. 15th National Conference on Artificial Intelligence (AAAI'98)*, 1998.

(Clarke *et al.*, 1999) E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

(Collins and Pryor, 1995) G. Collins and L. Pryor. Planning under uncertanity: Some key issues. In *Proc. 14th International Joint Conference on Artificial Intelligence*, 1995.

(Coradeschi and Saffiotti, 2000) S. Coradeschi and A. Saffiotti. Anchoring symbols to sensor data: Preliminary report. In *Proc. 17th National Conference on Artificial Intelligence*, 2000.

(Dean and Wellman, 1991) T.L. Dean and M.P. Wellman. *Planning and Control*. Morgan Kaufmann, 1991.

(Dechter *et al.*, 1991) R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61 – 95, 1991.

(DeCoste, 1990) D. DeCoste. Dynamic across-time measurement interpretation. In *Proc. 8th National Conference on Artificial Intelligence*, 1990.

(Despouys and Ingrand, 1999) O. Despouys and F. Ingrand. Propice-Plan: Toward a unified framework for planning and execution. In *Proc. 5th European Conference on Planning (ECP'99)*, 1999.

(Despouys, 2000) O. Despouys. *Une Architecture Intégrée pour la Planification et le Contrôle d'Exécution en Environnement Dynamique.* PhD thesis, l'Institut National Polytechnique de Toulouse, 2000.

(Doherty *et al.*, 2000) P. Doherty, G. Granlund, K. Kuchcinski, E. Sandewall, K. Nordberg, E. Skarman, and J. Wiklund. The WITAS unmanned aerial vehicle project. In *Proc. European Conference on Artificial Intelligence*, 2000.

(Emerson, 1990) E.A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, pages 997 – 1072. Elsevier, 1990.

(Forbes *et al.*, 1995) J. Forbes, T. Huang, K. Kanazawa, and S. Russel. The BATmobile: Towards a Bayesian automate taxi. In *Proc. 14th International Joint Conference on Artificial Intelligence*, 1995.

(Huang and Russel, 1997) T. Huang and S. Russel. Object identification in a bayesian context. In *Proc. 15th International Joint Conference on Artificial Intelligence*, 1997.

(Kaelbling *et al.*, 1998) L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99 – 134, 1998.

(Kalman, 1960) R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, March 1960.

(Koenig and Simmons, 1995) S. Koenig and R. Simmons. Real-time search in non-deterministic domains. In *Proc. 14th International Joint Conference on Artificial Intelligence*, 1995.

(Kowalski and Sergot, 1986) R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4, 1986.

(Matstoms *et al.*, 1996) P. Matstoms, H. Jönsson, and A. Carlsson. Beräkning av volume/delay-funktioner för nätverksanalys. *VTI Meddelande*, 777, 1996.

(McDermott, 1994) D. McDermott. An algorithm for probabilistic totally-ordered temporal projection. Technical Report YALEU/CSD/RR 941, Yale University, 1994.

(McIlraith *et al.*, 2000) S. McIlraith, G. Biswas, C. Clancy, and V. Gupta. Hybrid systems diagnosis. In *Proc. International Workshop on Hybrid Systems: Computation and Control*, 2000.

(McIlraith, 1998) S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proc. 6th International Conference on Principles of Knowledge Representation and Reasoning*, 1998.

(Mesarovic and Takahara, 1975) M.D. Mesarovic and Y. Takahara. *General Systems Theory: Mathematical Foundations.* Academic Press, 1975.

(Pasula *et al.*, 1999) H. Pasula, S. Russel, M. Ostland, and Y. Ritov. Tracking many objects with many sensors. In *Proc. 16th International Joint Conference on Artificial Intelligence*, 1999.

(Peot and Smith, 1992) M. Peot and D. Smith. Conditional nonlinear planning. In *Artificial Intelligence Planning Systems: Proc. International Confrence*, 1992.

(Puterman, 1994) M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, 1994.

(Rosen, 1991) R. Rosen. *Life Itself: A Comprehensive Inquiry into the Nature, Origin, and Fabrication of Life.* Columbia University Press, 1991.

(Saffiotti, 1998) A. Saffiotti. *Autonomous Robot Navigation: A Fuzzy Logic Approach.* PhD Thesis, Faculté de Science Appliquées, IRIDIA, Université Libre de Bruxelles, 1998.

(Smith and Weld, 1998) D. Smith and D. Weld. Conformant Graphplan. In *Proc. 15th National Conference on Artifical Intelligence (AAAI'98)*, 1998.

(Thiébaux and Lamb, 2000) S. Thiébaux and P. Lamb. Combining Kalman filtering and Markov localization in network-like environments. In *Proc. 6th Pacific-Rim International Conference on Artificial Intelligence*, 2000.

(Thrun, 2000) S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21, Winter 2000.

(Tijms, 1994) H.C. Tijms. *Stochastic Models.* Wiley, 1994.

(Tversky and Kahneman, 1974) A. Tversky and D. Kahneman. Judgement under uncertainty: Heuristics and biases. *Science*, 185:1124 – 1131, 1974.

(Weld *et al.*, 1998) D. Weld, C. Anderson, and D. Smith. Extending Graphplan to handle uncertainty & sensing actions. In *Proc. 15th National Conference on Artifical Intelligence (AAAI'98)*, 1998.

(Williams and Nayak, 1996) B.C. Williams and P.P. Nayak. A model-based approach to reactive self-configuring systems. In *Proc. 13th National Conference on Artificial Intelligence*, 1996.