# Integration of a control and a Predictive Maintenance System by a temporal Fuzzy-Controller

Thorsten W. Schmidt, Dominik Henrich

Embedded Systems and Robotics Lab. (RESY)
Chair for Applied Computer Science III
University of Bayreuth, D-95440 Bayreuth, Germany
E-Mail: {thorsten.w.schmidt,dominik.henrich}@uni-bayreuth.de
Http: //ai3.inf.uni-bayreuth.de

## Abstract

*A new approach for fuzzy control of control and maintenance systems is introduced. A standard fuzzy controller with additional and new predicates is used. These predicates handle temporal aspects to detect or predict the behaviour of a process in the past or in the future. With the ability to examine the past or future behaviour, a user of the temporal fuzzy logic controller can more easily integrate expert knowledge to describe the behaviour of a process. We will present an example of a room with an illumination system and cameras using such a fuzzy controller to prove our concepts.*

Keywords: temporal fuzzy logic, temporal fuzzy control language (TFCL), predictive maintenance system

## 1 Introduction

For many years, research has been done in the field of predictive maintenance and control systems. As an example, a renowned company in Germany working in the area of automatic milling machines recognised the value of predictive maintenance [Flender01, Flender02]. They provide special hardware to predict breakdowns in toothed wheels or vibrating parts. This system can be used on parts with a long life cycle for which the breakdown behaviour is well analysed, and thus represents a predictive maintenance system; but what is a predictive maintenance system? This question is answered in the following, step by step. The description begins with control systems and leads to maintenance systems via predictive control systems.

The *control system* observes the dynamic behaviour of a process with sensors inside the process. A control system is independent from an observed process. Deviations of sensor values from user-given values are recognised. The control system reacts to these deviations by alarming a user or trying to reduce the deviation with accessible parameters or actuators inside the machine [Castillo02]. This is the classical approach of a process controller. A simple example is a brightness control system for a room. If the room gets too dark, the brightness falls below a given threshold. Thus, more lamps have to be turned on to make the room brighter.

A *predictive control system* not only uses current sensor data from a process, it also predicts sensor data in advance and uses this data for the controller [Fantoni00] and [Palit00]. As any error that is detected in the predicted values might also occur in reality, it is thus the task of the controller to avoid the predicted faults. Since the controller can react much earlier to any possible fault, it is possible to completely avoid them. In the above example, this could mean that the room is slowly getting darker. Before the brightness falls below a user-given threshold, more lamps are turned on. In this manner, excessive darkening of the room is prevented.

A *maintenance system* is usually built upon a diagnosis system [Althoff92]. A diagnosis system is a failure driven system that waits for a malfunction and then starts a diagnosis to find the source of the failure. In this paper, the maintenance system is built upon a (predictive) control system. If a sensor value of a process is out of range and the control system is not able to change this state, this is called a non-adjustable fault in the process. Such a fault can occur because part of the process is damaged and must be replaced. In this case, the maintenance system generates a maintenance task for the user and tells the user which component of the process might be the source of this fault. With the predictive aspect of the maintenance system, it is possible to predict these faults before they occur. The process can run as long as needed, as the maintenance task can be scheduled for times when the process is under a low load or not running at all. This way the resulting downtime does not hinder the process as much as it would if the fault occurred at another time, because it is possible to combine several maintenance tasks. For the above-mentioned example, the scenario is the following: The room is getting darker and darker, while the predictive part of the controller predicts the required future light level. If all lamps are already turned on it is not possible to produce more light. At this point the maintenance system generates a maintenance task to replace the lights in this room, as some of them might be defective or producing less light than before.

The realisation of financial benefits and increased plant safety are the two main reasons for using maintenance systems [Kim99b]. It is not possible to separate these two aspects, because increased safety means reduced risk for loss of production, in turn leading to higher profit. To achieve higher safety, a small system with a low failure probability observes a large system with a relatively high probability of failure. [Ichtev01] stated that current predictive maintenance systems supervise only small components of airplanes, plants or robots, although it is desirable to supervise the complete system. The current predictive systems are customised for special tasks and work very well with those, but there is no flexibility to transfer this knowledge to other applications. For this reasons, there are only a few controllers for special applications with well known input and output behaviour.

With fuzzy control it is possible to describe the behaviour in a linguistic manner without knowledge of a detailed interrelationship or mathematical model. This description enables a user to easily insert his expertise into the controller. According to [Lepetic01], a fuzzy controller needs less time to converge than a PID controller with optimised parameters. Presently, such controllers are used to control processes, but we propose to use such a fuzzy controller as a maintenance system.

The Fuzzy Control Language (FCL, [IEC97]) provides a powerful standard for describing the behaviour of fuzzy controllers. This language allows users to specify all kinds of conditions for the fuzzy rules (unlike the Takagi-Sugeno rules [Takagi85], which can only handle the link **AND** between conditions).

A good review of fuzzy controllers is [Giron02], which summarises 200 papers discussing different types of fuzzy controllers. According to the reviewer, fuzzy controllers used for maintenance can be regarded as *predictive systems* in the category *stable fuzzy logic control alternatives*. Obviously, there is no class type for fuzzy controller used as predictive maintenance systems.

In another approach, based on predicate logic and a hierarchy in machine components, [Helmke99] developed a control system with a minimised knowledge base. Although it did not use fuzzy controllers, the benefits of fuzzy logic were discussed to cope with uncertainties in control and diagnostic knowledge.

In the past few years, new approaches to predict sensor data with fuzzy systems have arisen. The most elaborate method described is that of [Palit00], whereby the fuzzy rules were automatically learned, based on a sensor signal. These learned rules were used to make further predictions about this sensor signal.

It is possible to use temporal logic to describe future and past events in rules [Hajnicz96]. However, for special tasks such as maintenance, the two new predicates of temporal logic („*will be true in all future paths*" and „*will be true in one future path*") are not sufficient. Therefore the fuzzy controller needs to use some other form of temporal logic.

[Bovenkamp97] is a good introduction to temporal reasoning with fuzzy logic using fuzzy time-objects and uses fuzzy time-objects to represent uncertainty in both physical quantity and uncertain time. So, fuzzy time-objects are 3D membership functions. There are two reasons why we do not use this approach. First, the reasoning is done with just one predicate, but comparing predicate logic with temporal logic, we see that new predicates make it easier to understand and closer to natural language. Second, we are not interested in uncertainty in time. For our predicates we just want to know if they are true or false in a given time interval. Because of this, the reasoning with time is more easy.

In the rest of the paper, we developed this new form and present it in Section 2 where we describe the new predicates for the temporal fuzzy controller. Section 3 introduces a maintenance controller and explains its usage with fuzzy controllers. Section 4 presents a maintenance example, which is experimentally evaluated in Section 5. Lastly, Section 6 discusses the benefits of such controllers.

## 2 Temporal fuzzy logic

To control a general process about which only the input and output behaviour is known, the language usually used with fuzzy logic is insufficient. Normal fuzzy logic uses only the predicate **IS**, which allows an expert to check whether a process output has a specific fuzzy value, thus in this case generating an input for the process actuators. An example would be the rule „**IF** *input* **IS** *high* **THEN** *output* **IS** *low*". Note, the predicate **IS** (instead of „:=") is used to assign a fuzzy value to a fuzzy variable. But what about situations in which previous sensor data is used to describe a behaviour with a fuzzy rule? To use time-related data, a temporal enhancement is needed for the fuzzy controller.

To help an expert express his knowledge about time, he needs an easy-to-use language complex enough to describe this knowledge. A parser must analyse and understand this language, so spoken or written language is too complex and cannot be used. The goal is a language more complex than fuzzy logic; this is achieved by predicates introducing temporal aspects to fuzzy logic (Section 2.1), and by fuzzy rules that contain the necessary predicates (Section 2.2). We call this extension *Temporal FCL*.

## 2.1 Predicates

A predicate is a logical operation. It is used with arguments and, depending on the arguments, returns true (1.0) or false (0.0). In fuzzy logic, every value between 1.0 and 0.0 is allowed. Thus, a predicate can be regarded as a measurement of belief of its arguments.

Here, we have at least two arguments for each predicate. First, only predicates with two arguments are explained; predicates with more than two arguments are elucidated later. The first argument is a variable (e.g. *B*) of a known type (e.g. brightness) containing a value and the second is a fuzzy term for this type of variable (e.g. *veryLow* or *low*). The standard predicate in fuzzy logic is **IS**. With this information, we can build our first predicate with two arguments: *B* **IS** *veryLow*.

The predicate **IS** only regards the current state of a variable. To enhance this, we introduce future and past values for our variables. The values are represented by a variable (e.g. *B*(*t*) for our brightness) with a time index $t \in \mathbb{R}$ relative to the current time. A positive value implies a future value, while a negative value indicates a value in the past. For $i < j \, (i, j \in \mathbb{R})$ it is true that *B*(*i*) is an earlier value then *B*(*j*) The time stamps of the values are equidistant with a time step length $s \in \mathbb{R}$ , thus it is true for *i* and *j*: $\exists n \in \mathbb{N} | i = j + n \cdot s$ .

Past values are recorded and can be looked if needed. The future values are not yet known, so we need to predict these values. There are many methods for predicting time series (described in [Palit00] or [Jimenez92]), or standard mathematical methods such as the linear filtering (LF), moving average (MA) or autoregressive (AR) methods. Thus, we can assume that these future values are more or less accurate or at least they can give a hint, what might happen in the future. Of course, the accuracy of these values has a large effect on the methods presented later. More information about the accuracy can be found in [Iokibe00].

For protocols and model checking, [Karjoth87] presents an analyses for temporal predicates. He only uses temporal logic and not fuzzy logic, but he explained that four temporal predicates $\boxminus$ (was always true), $\diamondminus$ (was at least one time true), $\Box$ (will be always true), and $\Diamond$ (will be at least one time true) are sufficient to describe a process. Thus, our predicates for handling the past and the future are **WAS** ($\hat{=} \boxminus$), **PREEXIST** ($\hat{=} \diamondminus$), **WILL_BE** ($\hat{=} \Box$) and **WILL_EXIST** ($\hat{=} \Diamond$). The meaning of these predicates in combination with two arguments (one variable and one membership function) is described in the following Table 1.

| Predicate | Meaning |
|---|---|
| *B* **IS** *ft* | Checks how true the <u>current</u> value of $B(t)$ is for the fuzzy value *ft* and $t = 0$ |
| *B* **WAS** *ft* | Checks how true <u>the past</u> values of $B(t)$ were for the fuzzy value *ft* and $t < 0$ |
| *B* **PREEXIST** *ft* | Checks how true at least <u>one past</u> value of $B(t)$ was for the fuzzy value *ft* and $t < 0$ |
| *B* **WILL_BE** *ft* | Checks how true <u>the predicted</u> values of $B(t)$ will be for the fuzzy value *ft* and $t > 0$ |
| *B* **WILL_EXIST** *ft* | Checks how true at least <u>one predicted</u> value of $B(t)$ will be for the fuzzy value *ft* and $t > 0$ |

Table 1: Predicates comparing a variable $B(t)$ with fuzzy term *ft* at the time $t = 0$ (now), $t < 0$ (past) and $t > 0$ (future) The predicates can be used by a temporal fuzzy controller to handle past and future values.

Those four new predicates can have an additional argument: a time interval. Often, we are only interested in the values of a specific past or future time interval. Thus, the third argument is a time limit. Every value inside this limit is used to calculate the belief of the predicate. An example is *brightness* **WAS** *yesterday high*. Here, we are not interested if the brightness was high today or two days ago, we just want to know if it was high yesterday. In most cases a time interval is given because, for example, when using the predicate **WILL_EXIST** we can assume that the value of the variable will reach every possible value in the future, even if we must wait until eternity. Thus, the belief of **WILL_EXIST** without a limiting time interval is nearly 1.0.

## 2.2 Rules

Rules are used to describe the behaviour of a general process that may involve the use of many sensors and actuators. We therefore use the multiple-input-multiple-output (MIMO) approach for the fuzzy controller, thus enabling it to have several inputs and outputs. Furthermore, there is no restriction on the recursion depths of the conditions.

The following Table 2 briefly explains rule generation with Enhanced Baccus Naur Form (EBNF, [Scowen98]). The rules we can generate may look like the following example Rules:

- **IF** (*brightness* **PREEXIST** last minute *low*) **OR** (*brightness* **WILL_BE** next minute *high*) **THEN** *numberOfLamps*--

- **IF NOT** ((*brightness* **IS** *low*) **AND** (*brightness* **WAS** last hour *high*)) **THEN** *numberOfLamps++*

| **EBNF notation for rule generation** | |
|---|---|
| **IF** *Condition* **THEN** *ConclusionList* | |
| *Condition* | :== <[**NOT**] *Term***>** \| <[**NOT**] **(***Condition* {<**AND** \| **OR>** *Condition*}–**)**> |
| *ConclusionList* | :== (*Term*) \| <*ConclusionList*, *ConclusionList*> |
| *Term* | :== (*VarName Predicate FuzzyTerm*) \| (*varName*++) \| (*varName*--) |
| *Predicate* | :== <**IS** \| **WAS** *timeInterval* \| **WILL_BE** *timeInterval* \| **PREEXIST** *timeInterval* \| **WILL_EXIST** *timeInterval*> |
| *VarName* | :== *Name of a variable* |
| *FuzzyTerm* | :== *Name of a membership function (also called fuzzy term)* |
| *timeInterval* | :== *Values of a variable are taken from this time interval for the calculation of belief of a predicate* |

Table 2: EBNF table for building temporal fuzzy controller rules.

Every rule has a condition and a conclusion. The rule activation is calculated using the belief of the condition. The higher the belief of the condition, the higher the rule activation. After calculating the activation of all rules, only those with the highest degree of activation for a variable with the same fuzzy term will be used. With these activations the conclusion is activated. After that, the sharp output values are calculated for the fuzzy term activations.

So, using our predicates and rules, we can build up a rule base for the prediction example from Section 1. First, we need to have a rule (see Rule 1) that turns on more lights before the light level becomes too low, so we predict the brightness for the next quarter hour and turn on more lamps if the brightness will be low. If the brightness is high, we know that we can turn off lights (see Rule 2).

**IF** (*brightness* **WILL_BE** next quarter hour *low*) **THEN** *NumberOfLamps* ++ (Rule 1)

**IF** *(brightness* **IS** *high)* **THEN** *numberOfLamps– –* (Rule 2)

## 3 Maintenance Controller

The last section introduced the predicates and rules of the Temporal FCL. Next, we introduce the usage of maintenance tasks, which will always be in the form of user instructions, like „please replace defective light bulbs". The user knows what is to be done if he receives a specific instruction from the fuzzy controller. He receives no information about the process state; he is just given the probability and time for a possible maintenance task. The higher the belief for this event, the higher the probability that it will occur in the calculated time.

Besides the maintenance task event named *`maintenance`*, other events can occur. Every sensor has a maximum and a minimum warning level, and if the sensor value becomes larger than the maximum warning level (or smaller than the minimum warning level), a warning event named *`warning`* is produced. The error level encloses the warning level, i.e. the maximum error level is larger than the maximum warning level and the minimum error level is smaller than the minimum warning level. The error events are named *`error`*. If an error or warning occurs in the future, we call this a predicted error or predicted warning. These two events are called *`pError`* and *`pWarning`*, respectively. Figure 1 shows a sensor signal with its recorded values (left of current time) and its predicted values (right). The intervals of the warning and error level are shown additionally.

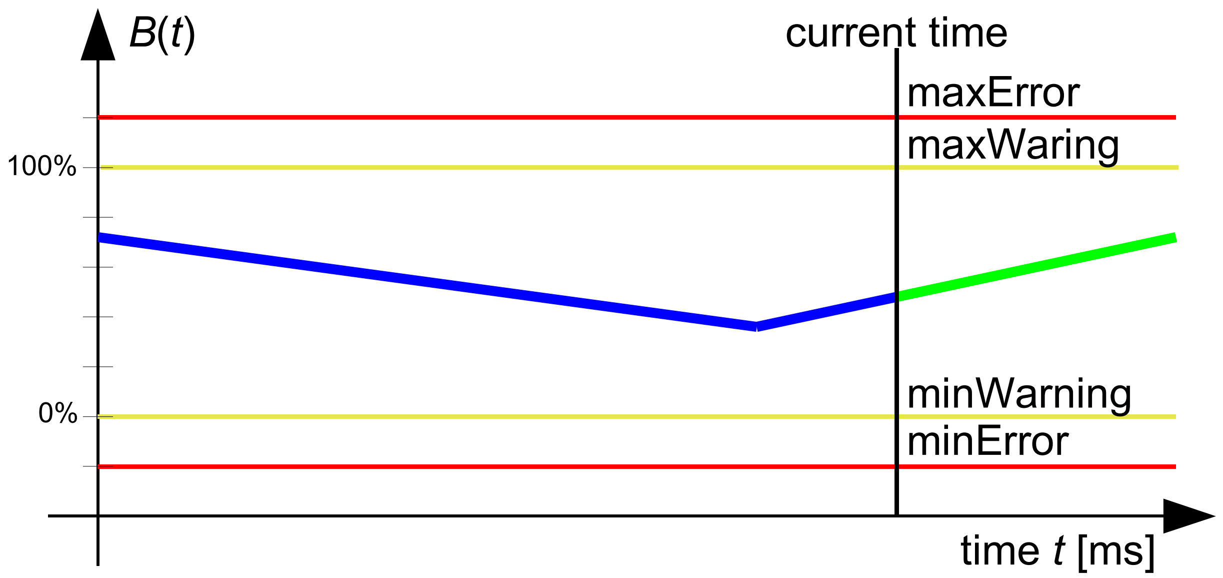

Figure 1: The vertical line represents the current system time. Recorded sensor data is to the left of current time, while predicted sensor data is to the right. The minimum and maximum warning and error levels are also given.

Now, we have five standard events (*`maintenance`*, *`warning`*, *`error`*, *`pWarning`* and *`pError`*) that can occur. The rules are intended to give the knowledge base developer an easy method to implement these events into the database. Every rule can have an output variable in its conclusion (see Section 2.2), the output variables being used to store the events. All possible tasks of one event type are combined as fuzzy terms in a variable (see task declaration of the event *`maintenance`* with several tasks $taks_i$ in Table 3). The different tasks are listed as singleton fuzzy terms with the names $task_i$. If a rule with such an output variable is activated, the given fuzzy term will store the rule activation. It is also possible that different tasks are activated by different (or the same) rules. If several rules activate this specific task, the highest activation will win. There will be no conflict, because every fuzzy term has its own activation. After all rules have fired, we can determine the probability of their occurrence using the different activations of the fuzzy terms. It is of course unnecessary to defuzzify this variable, because we are only interested in the activation of the fuzzy terms.

Table 3 shows how the developer declares a maintenance event handler for use in the knowledge base. Also shown are declaration of fuzzy terms $\texttt{task}_1$, $\texttt{task}_2$, ... representing the different maintenance tasks and declaration of a rule activating a maintenance task (here $\texttt{task}_1$). The condition can be any of those introduced in Table 2. The more the condition is activated, the more probable is the maintenance task.

| declaration | TFCL Code |
|---|---|
| **event** | `EVENT_VAR`<br>` maintenance`<br>`END_EVENT_VAR` |
| **task** | `EVENT maintenance`<br>` TERM task₁ := 1`<br>` TERM task₂ := 2`<br>` ...`<br>`END_EVENT` |
| **rule** | `RULE` *`n`*`: IF condition THEN maintenance := task₁` |

Table 3: Declaration of the maintenance event with tasks and its usage inside a rule.

When describing a process, some conditions occur more often and always lead to a maintenance event. In the following excerpt, the underlined conditions written in natural language stand exemplarily for such situations:

- The sensor signal is always between the minimum and maximum warning level, but one or more single values are too high or too low: **IF** future outlier is an error **THEN** maintenance **IS** $task_1$.
- The sensor signal is decreasing or increasing and will be out of range in the future: **IF** future trend leading to an error **THEN** maintenance **IS** $task_2$.
- After several unsuccessful tries to improve something there was no improvement: **IF** repeated improvement failure **THEN** maintenance **IS** $task_3$.

From the above examples, just the third example will be examined more closely. It is not possible to specify the condition repeated improvement failure in a single rule, so four rules expressing this condition are shown in Table 4.

| Rule types | Explanation |
|---|---|
| **IF** $condition_1$ **AND** $operation_i > -1$ **THEN** $operation_i$++, $A$ **IS** $ft$ | If $condition_1$ is true and the counter $operation_i$ is active (value > –1), we need to handle the situation with $A$ **IS** $ft$ that is executed. If we did this several times, there is need of a maintenance task. Therefore the counter $operation_i$ is increased by the rule activation. |
| **IF NOT** $condition_1$ **THEN** $operation_i$ – – | $A$ **IS** $ft$ is not executed, therefore $operation_i$ is reduced by the rule activation |
| **IF** $condition_2$ **THEN** $operation_i := 0$ | $Condition_2$ signalises that everything is fine again, so the counter $operation_i$ is reset to zero. |
| **IF** $operation_i > 5$ **THEN** $operation_i := -1$, maintenance **IS** $task_j$ | $A$ **IS** $ft$ was activated five or more times without advancement of $condition_1$, we deactivate the counter and generate a maintenance task. |

Table 4: Rules that occur often when a repeated improvement fails and leads to generation of a maintenance task.

## 4 Maintenance Example

This section describes a complete maintenance example written in TFCL (Temporal Fuzzy Control Language based on FCL). We use as an example a room with cameras that should be maintained at a minimum brightness and should not exceed a maximum brightness all the time. The constraint is that the lamps need about 15 minutes to reach their full brightness and it is too late to turn them on when the minimum brightness threshold is reached. Also the brightness should not exceed a maximum threshold. To prevent this we have shutter to dim the light from outside. Thus, we need to predict the brightness in the room. The brightness will change throughout the day because of dusk and dawn or clouds in front of the sun. If all eight lamps in the room are turned on, more than sufficient light is produced even if it is completely dark outside. And if all three shutter windows are closed, even the brightest sunlight can be reduce The TFCL file will, in addition to the maintenance section, include a controller to maintain the needed light in the room.

Table 5 includes the maintenance example written in TFCL. The system variables `numberOfLamps` and `schutterClosed` represent the number of lamps that should be turned on and number of shutters that should be closed, and so are special output variables. The input variable provides the measured brightness inside the room. The event variables `maintenance` and `pWarning` contain some of the events introduced in Section 3. The measured brightness of the room is fuzzified. We have five fuzzy terms: *veryLow*, *low*, *med*, *high* and *veryHigh*. A *veryLow* or *low* brightness inside the room is as undesirable as *high* or *veryHigh*, thus we would like a brightness value of *med*. This part is described in the first part of Table 5 (everything above the rule block).

After the declaration of variables and fuzzy terms, we have a rule block to monitor, predict maintain and control the system. The *control* and *prediction* rule (number 0) checks whether brightness is outside the range *veryLow* or *low*. If so, the predicted warning event `brightness` for the brightness is generated. The *maintenance* rule (number 1) checks if the brightness is since a quarter hour to low and more than 7 lamps are turned on. If so, the maintenance event *replaceLamps* is generated. The control rules (number 2-6) are used to turn lamps on or off or to open or close shutter. In detail, rule 2 turns on lamps, if it will be to dark in the next quarter hour and if all shutter are open. If a shutter is closed, we first want to open it, before we start to turn on lamps. Rule 3 opens a shutter, if the brightness will be to low in the next quarter hour. With the condition „*shutterClosed* **PREEXIST** last quarter_hour *shutterClosed*" we check if the number of closed shutter was constant in the past quarter hour. So, we only open a shutter if we have not opened or closed one in the last quarter hour. This is to avoid constant opening and closing of a shutter. Rule 4 and 5 turn off lamps if the brightness is to high and close shutter if all lamps are already turned off.

| Maintenance example in TFCL |
|---|

```
VAR_SYSTEM
 numberOfLamps actuator: Range: 0..8 REAL;
 shutterClosed actuator: Range: 0..2 REAL;
END_VAR
VAR_INPUT
 brightness: REAL;
END_VAR
VAR_EVENT
 maintenance;
 pWarning;
END_VAR_EVENT
FUZZIFY brightness
 TERM veryLow := (0, 1)(43, 1)(112, 0);
 TERM low := (43, 0)(112, 1)(128, 0);
 TERM med := (112, 0)(128, 1)(170, 0);
 TERM high := (128, 0)(170, 1)(213, 0);
 TERM veryHigh := (170, 0)(213, 1)(255,1);
 RANGE := (0 .. 255);
END_FUZZIFY
EVENT maintenance
 TASK replaceLamps := 1;
END_EVENT
EVENT pWarning
 TASK brightness := 1;
END_EVENT
RULEBLOCK
 AND:MIN;
 OR:MAX;
 ACCU:MAX;
 ACT:MIN;
 PREDICTION:LINEARITY;
 RULE 0: IF (brightness WILL_EXIST veryLow) OR (brightness
WILL_EXIST veryHigh) THEN (pWarning IS brightness);
 RULE 1: IF (brightness WAS last hour veryLow) AND
(numberOfLamps > 7) THEN (maintenance IS replaceLamps);
 RULE 2: IF ((brightness WILL_BE next quarter_hour veryLow)
OR (brightness WILL_BE next quarter_hour low)) AND
(shutterClosed < 0.5) THEN (numberOfLamps ++);
 RULE 3: IF ((brightness WILL_BE next quarter_hour veryLow)
OR (brightness WILL_EXIST next quarter_hour low)) AND
(shutterClosed PREEXIST last quarter_hour shutterClosed) THEN
(shutterClosed --);
 RULE 4: IF (brightness WILL_BE next quarter_hour high) OR
(brightness WILL_EEXIT next quarter_hour veryHigh) THEN
(numberOfLamps --);
 RULE 5: IF (brightness WILL_BE next quarter_hour high) OR
(brightness WILL_EXIST next quarter_hour veryHigh) AND
(numberOfLamps < 0.5) THEN (shutterClosed ++);
END_RULEBLOCK
```

Table 5: Complete TFCL File for control of room brightness and generation of maintenance tasks.

## 5 Experiment

This section describes a test of the example of Section 4 using the temporal fuzzy controller. The constraints were that the light level must not decrease to *low* or *veryLow* and it must not increase to *high* or *veryHigh*. The light is measured in the range 0 (minimum) to 255 (maximum). The threshold for changing the system variables *numberOfLamps* and *shutterClosed* is a rule activation above 50%. Thus, the rules 2-5 from Table 5 will only fire if the activation is more than 50%. With regard to the fuzzification and the brightness constraints of the room, this means the brightness inside the room will be regulated between 120 and 149 and should not exceed 170 or decrease 112.

The light *o* arising outside the room (influence from the sun) depends on the time of day *t* and is assumed to follow this curve:

$$o(t) = 192 \cdot \left(1 - \cos^2\left(\frac{t}{12h}\right)\right) + \text{noise}$$

If one of the eight available lamps is turned on, the brightness of the lamp increases linearly from 0 to 20 within 15 minutes.

To show the advantage of our controller with prediction, we use the controller with prediction turned off (see Figure 2) and with prediction turned on (see Figure 3) and compare the results with each other. The prediction is turned off by predicting always the last recorded value as future value. Thus, we can use for both tests the identical TFCL File, which makes the tests more comparable among each other.

The simulations without prediction show that the light level stays the most time of the day within the given interval. From 16:00 till 19:00 it is very low and at 19:15 it is even below the minimum brightness (112). The overage brightness is 135.6, the maximum 166.7, the minimum 110.5 and the standard deviation 7.20 (see Figure 2).

The simulations with prediction show that the light level stays all the time of the day within the given interval. The overage brightness is 138.7, the maximum 168.4, the minimum 122.0 and the standard deviation 7.93. (see Figure 3).

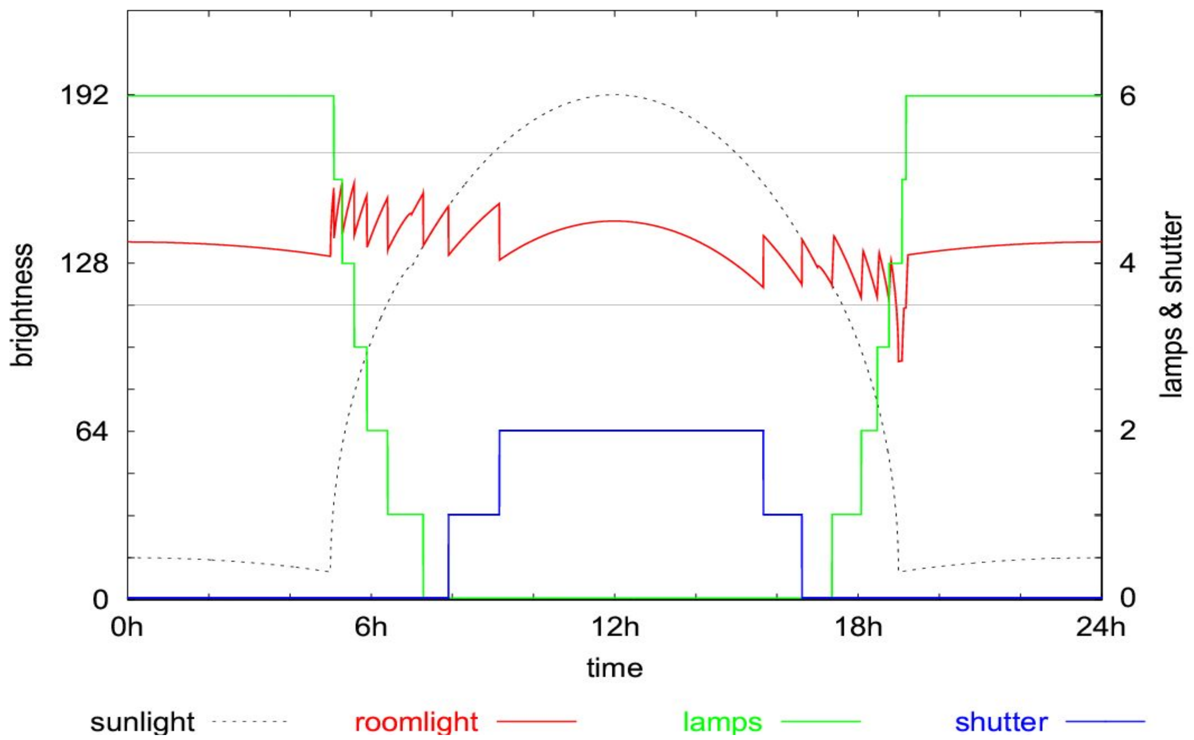

Figure 2: Experimental results without prediction for the distribution of lamps turned on (green), shutter closed (blue), light from outside (dotted line), and brightness inside the room (red) for a complete day (24 hours).

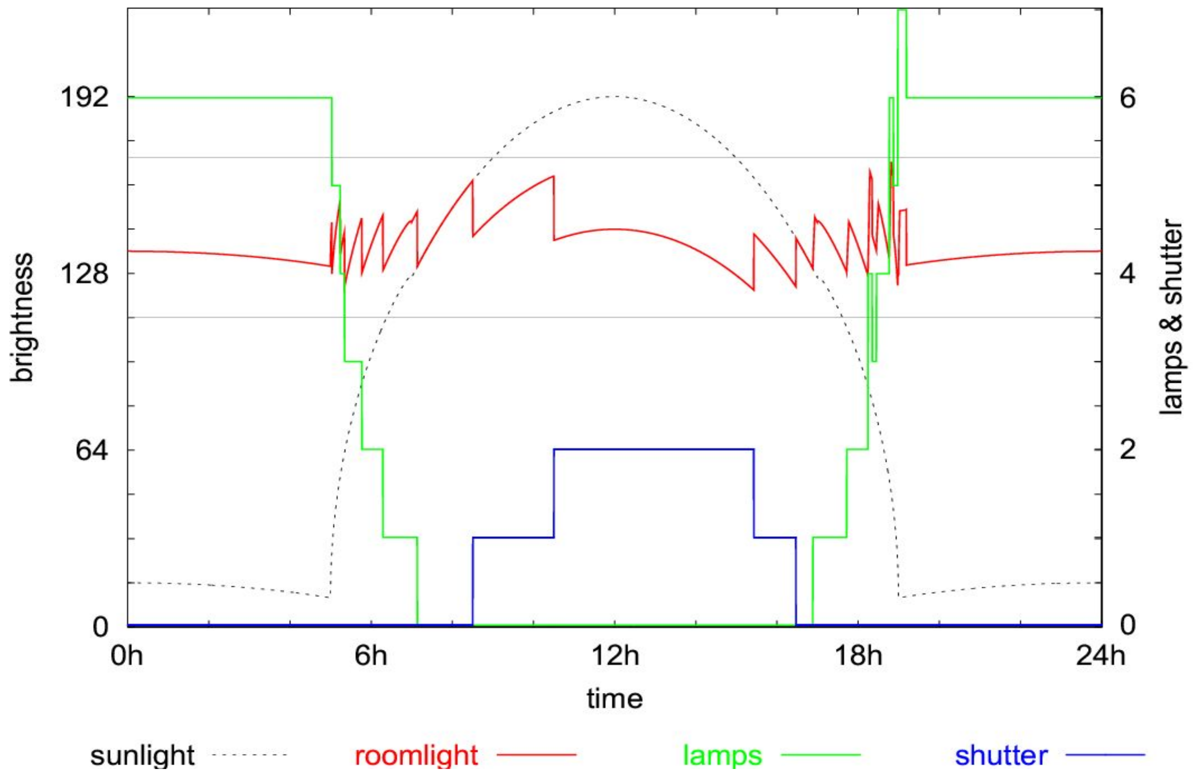

Figure 3: Experimental results with prediction for the distribution of lamps turned on (green), shutter closed (blue), light from outside (dotted line), and brightness inside the room (red) for a complete day (24 hours).

Compared with each other, the controller with prediction is better. The reason for this is, that the sunlight increases and decreases very fast. The decreasing is a problem for the controller without prediction, because if it detects that its to dark in the room, its to late to turn on the slow lamps. We need to turn them on earlier. Thus the controller with prediction can recognise this trend earlier and react much earlier on it. This can be seen in the example. The first lamp is turned on at 17:15 (Figure 2) and accordingly 16:45 (Figure 3).

## 6 Conclusions

The temporal fuzzy controller introduced uses new predicates to use sensor data from the past and future. The new language TFCL makes it possible to describe processes for which a standard fuzzy controller is unsuitable. However, this controller still lacks some features that would make it even more useful. For example, rule 5 from the example could be stated using (*brightness* **WILL_BE SMALLER** *high*), which is more compactly written and can be much more quickly evaluated by the controller. Thus, there are aspects of TFCL that could be improved to increase user friendliness. An additional way to make the software more user friendly is automatic generation of maintenance task schedules, keeping downtimes as low as possible.

As for the theoretical background, a complete mathematical model is required for the predicates presented. It also remains unclear what should be done with rules that contain different times in the condition; e.g. when should the conclusion be activated?

Also, we must perform benchmark tests of all the possible predicates and prediction functions of the fuzzy controller to emphasise that these examples are not possible with a standard fuzzy controller. Currently, the software prototype is being integrated in the vision system of the room to observe false pixels and to initiate a recalibration of the system.

## 7 Acknowledgments

This research project has been supported by the European Community in the field of research, technological development and demonstration (RTD) for the Fifth Framework Programme in the Specific Programme „Promoting Competitive and Sustainable Growth" (No. G1RD-CT2000-00386).

Also, we want to thank Thomas Carl for grammatical proofreading of this paper.

## 8 References

[Althoff92] K.-D. Althoff, „Eine fallbasierte Lernkomponente als integrierter Bestandteil der MOLTKE-Werkbank zur Diagnose technischer Systeme", Dissertation, Kaiserslautern, September 1992

[Bothe95] H.-H. Bothe, „Fuzzy Logik – Einführung in Theorie und Anwendungen", Springer-Schoolbook, 2. Edition, Berlin Heidelberg, 1995

[Bovenkamp97] E. G. P. Bovenkamp, J. C. A. Van der Lubbe, „Temporal reasoning with fuzzy time-objects", 4th International Workshop on Temporal Representation and Reasoning (TIME '97), 10-11 May 1997

[Camacho95] E. F. Camacho, C. Bordons, „Model predictive control in the process industry", Springer Press, London, 1995

[Castillo02] O. Castillo, P. Melin, „A New Approach For Plant Monitoring Using Type-2 Fuzzy Logic and Fractal Theory", Proceedings of the 5th International FLINS Conference, Belgium, September 2002

[Fantoni00] P.F. Fantoni, M. Hoffmann, B. H. Nystad, „Integration of sensor validation in modern control room alarm systems", Proceedings of the 4th International FLINS Conference, Belgium, August 2000

[Flender01] Flender Service GmbH, „Condition Monitoring - Maschinendiagnose, GearController und Antriebsservice", www.flender-cm.de/deutsch/upload/aboutesat.htm, March 2001

[Flender02]Flender Service GmbH, „Condition Monitoring for the highest Availability of Power Technology”, http://www.flender-cm.de/images/pdffiles/Leistungskurzschreibung_GB.pdf, June 2002

[Giron02] J. M. Giron-Sierra, G. Ortega, „A Survey of Stability of Fuzzy Logic Control with Aerospace Applications”, IFAC Proceedings of the 15th Triennial World Congress, Bacelona, Spain, 2002

[Hajnicz96] E. Hajnicz, „Time structures: formal description and algorithmic representation”, Berlin Heidelberg, Springer Press, 1996.

[Helmke99] H. Helmke, „Ein wissensbasiertes Modell für die On-line-Überwachung und – Diagnose technischer Systeme”, Deutsches Zentrum für Luft- und Raumfahrt e.V., Braunschweig, 1999

[IEC97] IEC TC65/WG 7/TF8, ”Fuzzy Control Programming”, International Technical Electronical Commission (IEC), 1997

[Ichtev01] A. Ichtev, J. Hellendoorn, R. Babuška, „Fault Detection and Isolation Using Multiple Takagi-Suego Fuzzy Models”, Proceedings of the International Fuzzy Systems Conference, Melbourne, Kanada, December 2001

[Iokibe00] T. Iokibe, M. Koyama, M. Taniguchi, „A Study for Complexity of Chaotic Time Series and Prediction Accuracy”, Proceedings of the International FLINS Conference, Belgium, August 2000

[Jimenez92] J. Jimenez, J. Moreno, G. Ruggeri, „Forecasting on Chaotic Time Series: A Local Optimal Linear reconstruction Method”, Physical Review A, Vol. 45, No. 6, pp. 3553-3558, 1992.

[Karjoth87]G. Karjoth, „Prozeßalgebra und temporale Logik – angewandt zur Spezifikation und Analyse von komlexen Protokollen”, Diss. Mathematik/Informatik, Universität Stuttgart, 1987

[Kim99b] H. Kim, S. Moon, J. Choi, S. Lee, D. Do, M. M. Gupta, „Generator Maintenance Scheduling Considering Air Pollution Based on the Fuzzy Theory”, IEEE International Fuzzy Systems Conference, Proceedings, pp. 1759-1764, August 1999

[Lepetič01]M. Lepetič, I. Škrjanc, H. G. Chiacchiarini, D. Matko, „Predictive Control based on Fuzzy Model: A Case Study”, Proceedings of the International Fuzzy Systems Conference, Melbourne, Canada, December 2001

[Mees91] A. Mees, „Dynamical Systems and Tessellations: Detecting Determinism in Data”, International Journal of Bifurcation and Chaos, Vol 1, No. 4, pp. 777-794, 1991.

[Palit00] A. K. Palit, „Artificial Intelligent Approaches to Times Series Forecasting”, Dissertation, Bremen, January 2000

[Schwarz97] H. R. Schwarz, „Numerische Mathematik”, Teubner, 4. Edition, Stuttgart, 1997

[Scowen98] R. S. Scowen, „Extended BNF – A generic base standard”, Great Britain, 1998

[Takagi85] T. Takagi, M. Suego, „Fuzzy identification of systems and its application to modelling and control”, IEEE Transactions on Systems, Man and Cybernetics, Vol. 15, No. 1, pp. 116-132, 1985.

Dipl.-Inform. **Thorsten W. SCHMIDT** received the Diploma (Master's Thesis) in Computer Science from the University of Kaiserslautern, Germany, in September 2001. During September 2001 to July 2003, he worked for the Workgroup Robotics and Embedded Systems (RESY) in the project "Vision assisted machine for recycling applications (VISREC)" at the University of Kaiserslautern, Germany. Since August 2003, he works in the Chair for Applied Computer Science III (Robotics and Embedded Systems) at the University of Bayreuth, Germany. His fields of investigation and interest are temporal fuzzy control, industrial and mobile robotics, process simulation and artificial intelligence.

Prof. Dr. **Dominik HENRICH** obtained his Diploma (Master's Thesis) in Computer Science in 1991 from the University of Karlsruhe, Germany. During 1992 to 1994, he was supported by a stipend granted from the German Research Foundation (DFG) and finished his Doctorate (Ph.D. Thesis). During 1996 to 1999, Prof. Henrich built up the research group for "Parallel Processing and Robotics" at the Insitute for Process Control (IPR) and was lecturer at the Informatics Faculty of the University of Karlsruhe. In 1998, he received a STA fellowship at the Electrotechnical Laboratory of the Ministry of International Trade and Industry (MITI), Japan. During 1999 to 2003, he headed as Professor the research group "Embedded Systems and Robotics (RESY)" at the Informatics Faculty of the University of Kaiserslautern. Since August 2003, he holds the chair for Applied Computer Science III (Robotics and Embedded Systems) at the University of Bayreuth. The research interests of Prof. Henrich are robot simulation and programming, robot manipulation skills, force- and vision-based manipulation, handling deformable objects, automatic motion planning, collision detection and avoidance.