Alarm Analysis with Fuzzy Logic and Multilevel Flow Models

Fredrik Dahlstrand
Department of Information Technology
Lund Institute of Technology
Box 118, SE-221 00 Lund, Sweden
Phone: +46 46 222 95 10
Fax +46 46 222 47 14
E-mail: fredrikd@it.lth.se

Abstract

This paper presents one method for combining Multilevel Flow Models (MFM) alarm analysis and fuzzy logic. The already existing MFM alarm analysis algorithm, which is using discrete logic, has several problems when confronted with uncertainty such as noise or signals close to a decision limit. The new result presented in this paper is a fuzzy MFM alarm analysis algorithm, which is more reliable when faced with uncertainties.

1 Introduction

In many large industrial processes a simple alarm situation may turn into a complex one due to the huge amount of information that is presented to a human operator. The difficulty, when performing alarm analysis, is not to obtain the information but rather to sort out which information is important in a certain alarm situation.

One way to reduce the amount of information presented is to model the causality of the process, that is, to describe which components depend on each other. With such a model it is possible to determine the root cause of an alarm situation because it is possible to find out which alarms are consequences of other alarms. Professor Morten Lind has introduced a way to model consequential dependencies in a process: Multilevel Flow Models [6], from now on referred to as MFM.

MFM is used to model the intentions of a man-made (technical) process and the means by which these intentions are achieved. Further work by Larsson has introduced a reliable method to perform an alarm analysis [2, 3]. MFM alarm analysis has three main advantages:

1. It is a relatively easy task, compared to expert systems, to build, update and maintain the knowledge base, this due to the graphical nature of MFM. Furthermore, the high level of abstraction requires less detailed knowledge of the system.

2. With the already existing alarm analysis algorithm it is possible to accurately measure the worst-case execution time.
3. The already existing MFM algorithm is very fast. The worst case execution time for the Guardian model [5], which corresponds to 374 backward chaining rules, is about 7000 diagnoses per second. The test was performed on a 200 MHz Pentium Pro running the Windows NT operating system.

MFM fault diagnosis and alarm analysis have successfully been tested on real world processes. Larsson [2] describes how MFM was tested on the Steritherm process. The Steritherm process is used to sterilise liquid food products, for example, milk. In the Guardian project [5], MFM was used to monitor the human body, and more specifically, to monitor a patient in an intensive care unit. In an ongoing project several solutions to combine MFM with quantitative and probabilistic methods are investigated.

This method has however one drawback, it does not handle uncertainties in the process, such as signals close to decision limit, noisy signals, missing data, etc. The already existing algorithm uses crisp logic with two or three distinct values, for example, the flow through a pipe may either be described as too low, too high or normal. A situation, as described in Figure 1, may cause a burst of random alarms. The signal is likely to be within limits (the horizontal lines show the desired interval) but due to noise the alarm analysis switches chaotically between failed and working states.

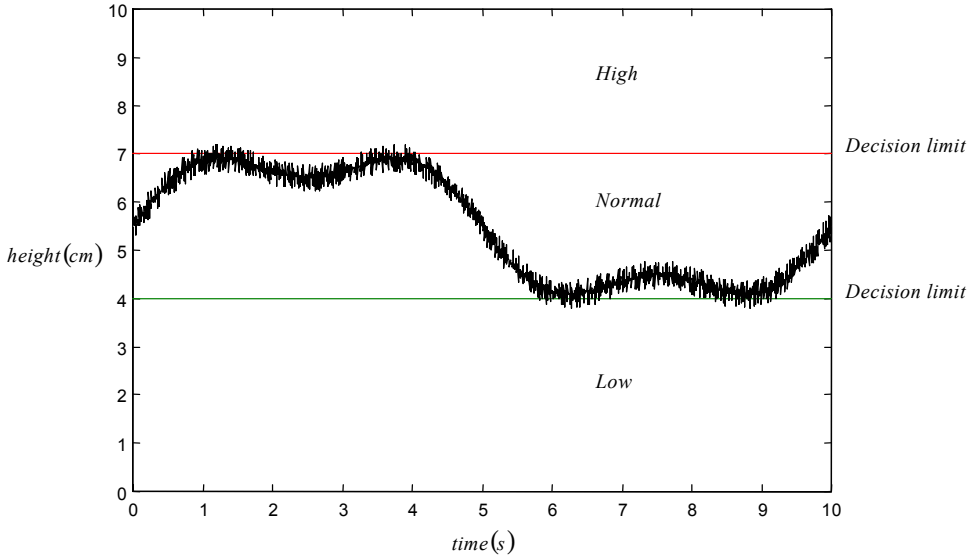

Figure 1: *A noisy signal close to decision boundaries. The desired signal level is between the two horizontal lines.*

Ideally, in a situation as the one in Figure 1, the algorithm should not indicate an alarm but rather indicate that the signal is close to a decision limit.

This paper shows one approach to combine the concept of MFM alarm analysis and fuzzy logic to produce a more reliable alarm analysis.

2 Multilevel Flow Models

MFM models are graphical representations of the goals and functions of technical systems. The goals describe the purposes of the system, and the functions describe the capabilities of the system. MFM models describe the functional structure of a system by modelling the flow structures in the system. MFM models are built by using three basic concepts, *goals*, *functions*, and *physical components*.

2.1 Functions

MFM describes the functionality of a system by modelling the flows of *mass*, *energy* and *information*. The mass, energy, and information flows are modelled by using six different flow functions:

- The *source* function is used to model the capability of providing an infinite amount of mass, energy, or information.
- The *sink* function is used to model the capability of receiving an infinite amount of mass, energy, or information.
- The *transport* function is used to model the capability of transporting mass, energy, or information.
- The *barrier* function is used to model the capability of preventing mass, energy, or information transport.
- The *storage* function is used to model the capability of storing mass, energy, or information.
- The *balance* function is used to model the capability of forking and combining different flows of mass, energy, or information. The flows must be of the same basic type.

Beside these six functions, information flows may be modelled using three more functions:

- The *observer* function is used to model the capability of monitoring the state of a physical component and translate this into a flow of information.
- The *actor* function is used to model the capability of altering the state of a physical component.
- The *decision* function is used to model the capability of decision making. Decisions may be made either by control algorithms or by human operators.

There exists one more function, the *manager*, which is used to model the capability of resource management and control. The manager is used to describe a

system that is intended to manage a certain task. The graphical representations of the MFM functions are shown in Figure 2.

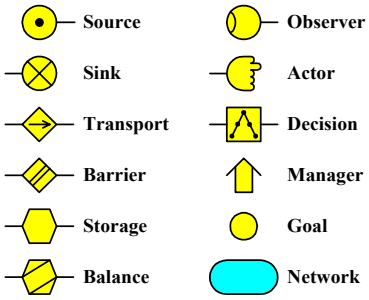

Figure 2: *The graphical representation of functions, goals, and networks.*

Functions may be connected to each other into flow paths. However, the flow functions may not be connected to each other in an arbitrary manner. The legal MFM connections are shown in Figure 3. For example, the storage must be connected to either a transport or a barrier at each connection point. A balance must be connected to at least one transport or barrier at each end. The observer, sink, source, and actor have only one connection point, thus acting as the end or the beginning of a flow structure.

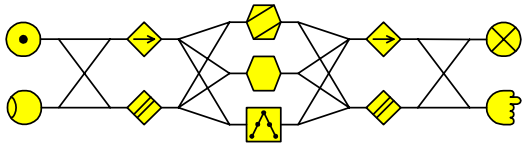

Figure 3: *All legal MFM function to function connections. In this figure the functions have been place so that a function may only be connected to functions to the left or to the right, but not to functions above or below.*

2.2 Relations

The goals, functions, and physical components in an MFM model are dependent on each other. To model this, four different types of relations are used:

- An *achieve* relation is used to connect a group of MFM functions that work together to fulfil the connected goal.
- An *achieve-by-management* relation is the same as an *achieve* relation with the extension that the goal is fulfilled by management of resources. The management is done by, for example, human operators or PID-controllers.
- A *condition* relation is used to model that the connected goal must be fulfilled to allow for the connected function to be available.

- A *realise* relation is used to connect the physical components to the functions. This is normally not shown in an MFM model.

The graphical representation of these relations is shown in Figure 4.

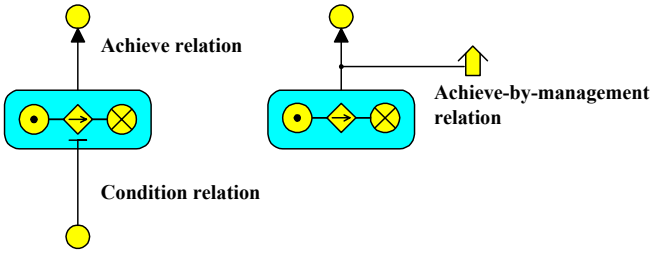

Figure 4. *The MFM relations.*

2.3 An Example of an MFM Model

The following example, the heat exchanger in Figure 5, is a part of the Steritherm [2] process. Water is pumped through a steam injector, where it is heated with steam. The heated water is then transported through a plate heat exchanger and heats the product to the desired temperature.

This process has three goals:

G1 Heat the product to desired temperature.
G2 Bring product to heat exchanger.
G3 Bring water to heat exchanger.

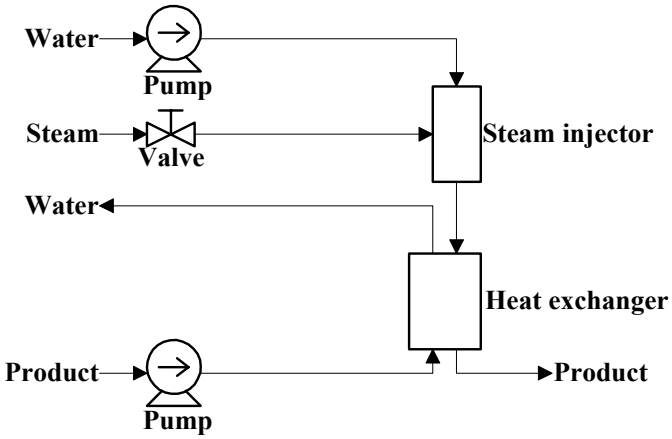

Figure 5: *Process schematics of a heat exchanger.*

G1 is the primary goal of the process. G2 and G3 are subgoals, which are required in order to achieve G1.

Even though the process is rather small there are many functions present, as seen in Table 1.

	Function	The function is realised by
F1	Provide thermal energy.	The steam system.
F2	Transport thermal energy.	The steam valve.
F3	Provide thermal energy.	The water tank.
F4	Transport thermal energy.	The water tank.
F5	Transport thermal energy.	The steam injector.
F6	Transfer thermal energy between media.	The heat exchanger.
F7	Receive thermal energy.	The product.
F8	Provide steam.	The steam system.
F9	Transport steam.	The steam valve.
F10	Provide water.	The water tank.
F11	Transport water.	The water pump.
F12	Mix water and steam.	The steam injector.
F13	Transport heated water.	The heat exchanger.
F14	Receive heated water.	The water tank.
F15	Provide product.	The product tank.
F16	Transport product.	The product pump.
F17	Receive product.	Another part of the Steritherm process where the packing of the product takes place.

Table 1: *The functions in the Steritherm process.*

The third basic concept of MFM is the physical components. In this process the physical components are:

C1 Product tank.
C2 Product pump.
C3 Heat exchanger.
C4 Water tank.
C5 Water pump.
C6 Steam system.
C7 Steam valve.
C8 Steam injector

The MFM model of the process in Figure 5 is shown in Figure 6. The dots below some of the functions indicate that a sensor in the process monitors those functions' alarm states. F2, F4, and F6 have temperature sensors, which measure the temperature of the steam, the water, and the product respectively. F9, F11, F13,

and F16 have flow sensors, which measure the flow of the steam, the water, the steam/water mixture, and the product respectively.

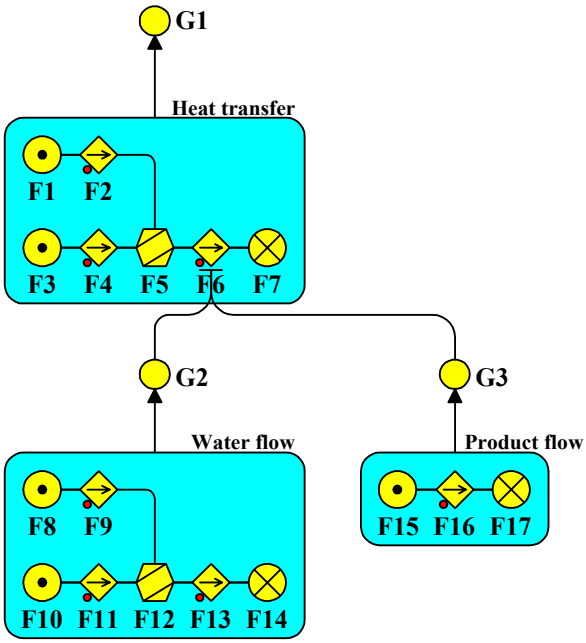

Figure 6: *An MFM model of the heat exchanger in Figure 5.*

2.4 Alarm Analysis

Even though the previously described process is small, it contains several sensors. A single failure may cause consequential failures, for example, if the steam system does not provide enough steam then following alarm situation may occur:

- F9 reports that the flow of steam is too low.
- F2 reports that the temperature of the steam is too low.
- F6 reports that the heated product is not hot enough.

In the situation described above it may not be too hard to figure out which alarm is the root cause. However, in a process with hundreds of sensors, the alarm situation may not be as easily handled. The already existing MFM alarm analysis algorithm introduced by Larsson [2, 3] handles alarm situation by organising the alarms into two classes:

1. *Primary alarms*. These are the alarms that are directly connected to the primary source of the failure.

2. *Secondary alarms*. These alarms may be the consequence of a primary alarm. However, it is also possible that they are hidden primary alarms, that is, primary alarms that are failed in such way that they appear to be caused by another failure.

Each of the MFM functions has a set of alarm states that indicates in which way they have failed. These are described in Table 2.

Alarm state	Description
locap	The inflow, or outflow, is greater than intended for the function.
loflow	The flow through the function is lower than intended.
hiflow	The flow through the function is higher than intended.
lovol	The stored volume in the function is lower than intended.
hivol	The stored volume in the function is higher than intended.
leak	The inflow is greater than the outflow.
fill	The outflow is greater than the inflow.

Table 2: *Description of the MFM alarm states.*

Each function may only be in some of these states. The possible alarm states for each function are enumerated in Table 3. Besides the alarm states each function may be in a normal state, that is, working as expected.

Function	Alarm states
Source	*locap, normalcap*
Transport	*loflow, normalflow, hiflow*
Barrier	*leak, normal*
Storage	*lovol, normalvol, hivol, fill, normal, leak*
Balance	*loflow, normalflow, hiflow, fill, normal, leak*
Sink	*locap, normalcap*

Table 3: *The possible alarm states for the different MFM functions.*

Figure 7 shows a part of the process in Figure 5. Assume that the water pump (F11) transports too much water. Even though the water tank is able to provide as much water as the pump requires, it will eventually run out of water, thus being in

a state of *locap*. Since the pump is the cause of the *locap* alarm, the transport (F11) is guessed to be primary, and (F10) is guessed to be secondary.

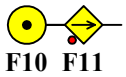

Figure 7: *An MFM alarm situation.*

This type of intelligent guessing in the MFM alarm analysis algorithm is called *consequence propagation*. Table 4 lists the rules for the consequence propagation used in the example above. The source is denoted by S and the transport is denoted by T. Part A shows the rules for alarm state consequence propagation. Note that this only applies to functions without sensors. Part B shows the rules used to determine the failure state of the source function, and finally part C shows the rules used to classify the alarm. Similar rule sets exist for all legal MFM connections, see Larsson [2, 3].

A	$\neg hiflow(T) \Rightarrow normalcap(S)$ $hiflow(T) \Rightarrow locap(S)$
B	$normalcap(S) \Rightarrow working(S)$ $locap(S) \Rightarrow failed(S)$
C	$normalcap(S) \Rightarrow ok(S)$ $\neg hiflow(T) \wedge locap(S) \Rightarrow primary(S)$ $hiflow(T) \wedge locap(S) \Rightarrow secondary(S)$

Table 4: *The rules used for the consequence propagation between a transport (T) and a source (S).*

To describe how the flow structure is searched when performing an alarm analysis, the example MFM model in Figure 8 will be used. The dots indicate the functions that are measured (connected to sensors in a process). If the transport (F11) is the most recently changed alarm, then the algorithm starts by searching all possible paths through the flow structure in such a way that every function appears only once in each path. The search ends when a sensor is found, or when a function with only one connection point is reached, that is, a source, an observer, a sink or an actor function.

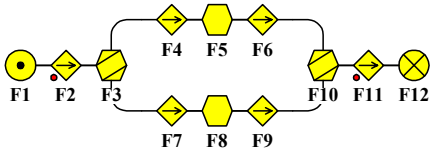

Figure 8: *An example of a MFM alarm analysis situation.*

The tree is traversed according to the following algorithm, the Waltz algorithm [4]:

1. Search downward for a leaf node.
2. If the leaf node is a function with a sensor that reports an alarm, perform the consequence propagation back to the root node.
3. If the root node is now considered a primary alarm for this path then perform the consequence propagation down to the leaf node.
4. Search downward for a new leaf node and repeat from 2 until all paths have been analysed.

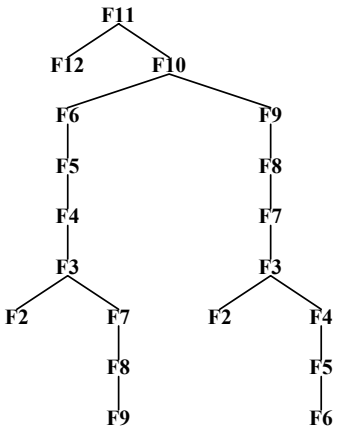

Figure 9: *The tree searched when performing an alarm analysis on the MFM model in Figure 8.*

Using this algorithm, two types of conflicts may occur:

- A function's guessed alarm state may be high based on one path, or low based on another path. This problem is solved by introducing a new guessed alarm state, *hilo*, which indicates that the function's alarm state is either high or low.
- The function at the root may be considered a primary alarm based on one path, or secondary based on another path. This problem is solved by assuming that the function's alarm is secondary, since it is probably caused by another alarm. Even if it actually is primary the definition of a secondary alarm states that it might be a hidden primary alarm.

3 Fuzzy Alarm Analysis

The crisp logic MFM analysis algorithm has some problems when faced with noisy signals, and signals close to a decision limit. Fuzzy logic [7] would be an interesting approach to come to terms with some of the problems concerning the

uncertainties in the real world. The following benefits are gained when combining the existing MFM alarm analysis with fuzzy logic:

- The alarm analysis algorithm is more reliable when there are disturbances caused by small and rapid changes.
- It is possible to grade the closeness to a decision limit.
- It is possible to grade the failure, that is, how failed a function is.
- The rules and the algorithm of the already existing MFM alarm analysis algorithm may be used.

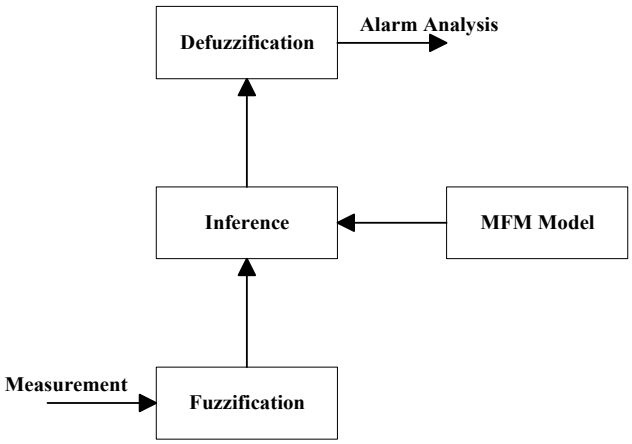

Figure 10: *An architecture for fuzzy MFM alarm analysis.*

Figure 10 shows the architecture used for fuzzy MFM alarm analysis. First the measured process values are fuzzified. Then the connections in the MFM model determine which rules are used by the inference. Finally, the results of the fuzzy alarm analysis are defuzzified back to information, which is presented to the operator or some higher level diagnostic algorithm.

Throughout the remainder of the section the fuzzification, the inference, and the defuzzification stages will be described. To demonstrate the proposed fuzzy analysis algorithm, the following alarm situation is used (see Figure 5 and Figure 6):

- F9 reports that the flow of steam is too low.
- F13 reports that the flow of the steam/water mixture is too low.

Figure 11 shows the search tree for the alarm situation above. To limit the example, the path F9–F12–F13 is the only path that will be analysed.

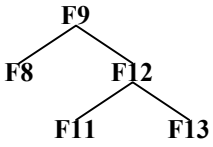

Figure 11: *The search tree for the alarm situation.*

3.1 Fuzzification

To transform the measured value into a fuzzy alarm state, fuzzy sets as those in Figure 12 are used. Figure 12 shows the *loflow*, *normalflow*, and the *hiflow* fuzzy sets used for the alarm states for transports and balances. For each of the other alarm states similar fuzzy sets exist.

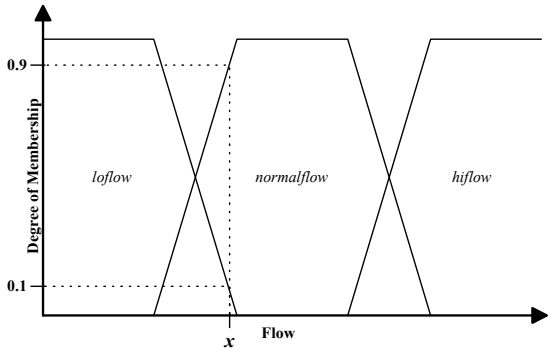

Figure 12: *The fuzzy sets used when determining the alarm state of a transport or a balance.*

Assume that the flow of the steam/water mixture through the heat exchanger (F13 in Figure 6) is x. Then the alarm state of the function belongs to the fuzzy set *loflow* to a degree of 0.1 as shown in Figure 12. The grades of membership to the other fuzzy sets are obtained in the same way, see Table 5. F9's alarm state belongs more to the *loflow* fuzzy set than to the *normalflow* fuzzy set, while F13's alarm state belongs more to the *normalflow* fuzzy set than to the *loflow* fuzzy set.

Function	*loflow*	*Normalflow*	*hiflow*
F9	0.9	0.1	0.0
F13	0.1	0.9	0.0

Table 5: *Degree of membership to the alarm state fuzzy sets for the measured functions.*

3.2 Inference

The inference engine uses the same rules as the already existing discrete logic MFM alarm analysis algorithm, see Table 4. Using the steps of the Waltz algorithm the steps will be:

1. The search starts at F9 and ends at F13 because F13 has a sensor.
2. Since the sensor of F13 reports an alarm the consequence propagation is done according to the consequence propagation rules from F13 to F9. The result is shown in Table 6.

	F13	F12	F9
loflow	0.1	0.1	0.9
normalflow	0.9	0.9	0.1
hiflow	0.0	0.0	0.0
working	0.9	0.9	0.1
failed	0.1	0.1	0.9
ok	0.9	0.9	0.1
primary	0.1	0.0	0.0
secondary	0.0	0.1	0.1

Table 6: *The result of the F13–F12–F9 consequence propagation.*

3. Since the sensor of F13 also is in *normalflow* state, and thus F9 must be a primary alarm, the consequence propagation is done from F9 to F13. The result is shown in Table 7.

	F9	F12	F13
loflow	0.9	0.9	0.1
normalflow	0.1	0.1	0.9
hiflow	0.0	0.0	0.0
working	0.1	0.1	0.9
failed	0.9	0.9	0.1
ok	0.1	0.1	0.9
primary	0.9	0.0	0.0
secondary	0.0	0.9	0.1

Table 7: *The result of the F9–F12–F13 consequence propagation*

Now the results of step 2 and 3 must be merged together. This is done according to the rules in Table 8.

$loflow(step2) \vee loflow(step3) \Rightarrow lowflow(final)$
$normalflow(step2) \wedge normalflow(step3) \Rightarrow normalflow(final)$
$hiflow(step2) \vee hiflow(step3) \Rightarrow hiflow(final)$
$working(step2) \wedge working(step3) \Rightarrow working(final)$
$failed(step2) \vee failed(step3) \Rightarrow failed(final)$
$ok(step2) \wedge ok(step3) \Rightarrow ok(final)$
$primary(step2) \vee primary(step3) \Rightarrow primary(final)$
$secondary(step2) \vee secondary(step3) \Rightarrow secondary(final)$

Table 8: *The rules for merging together step 2 and step 3 of the Waltz algorithm.*

Here, *step2* is the function's calculated fuzzy values in step 2, *step3* is the function's calculated fuzzy values in step 3, and *final* is the result of the consequence propagation. The result of the consequence propagation is shown in Table 9.

	F9	F12	F13
loflow	0.9	0.9	0.1
normalflow	0.1	0.1	0.9
hiflow	0.0	0.0	0.0
working	0.1	0.1	0.9
failed	0.9	0.9	0.1
ok	0.1	0.1	0.9
primary	0.9	0.0	0.1
secondary	0.1	0.9	0.1

Table 9: *The final result of the consequence propagation.*

3.3 Defuzzification

After the alarm analysis has been performed the desired knowledge is "how failed the function is", and whether it is a cause or a consequence, that is, primary or secondary. To obtain this knowledge from the fuzzy values in Table 9 the fuzzy sets in Figure 13 are used. For the consequence state a similar set is used.

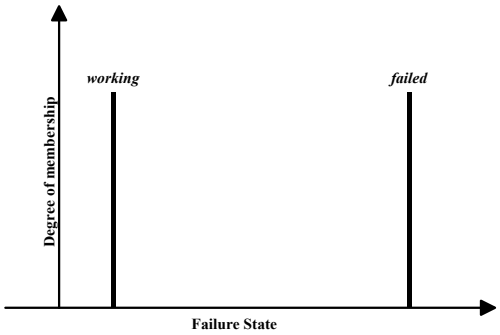

Figure 13: *The fuzzy sets for the failure state.*

The defuzzication is performed by using the centre of gravity method [1]. Figure 14 shows the defuzzication of the consequence state for function F9.

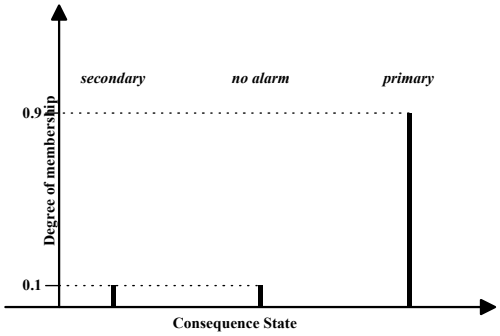

Figure 14: *Example of defuzzication using the result of the consequence propagation in Table 9 and the consequence state fuzzy sets.*

The output from the defuzzifier is presented as two numbers. The first number indicates how failed the function is, and the second number indicates if the alarm is primary or secondary. In this example the value of failure ranges from 0 to 100, where 0 means working, and 100 means that the function is failed. Furthermore, the consequence value ranges from –1 to 1, where –1 means a secondary alarm, 1 means a primary alarm, and 0 means working as expected (no alarm situation). These ranges may be chosen to suit the needs of higher level diagnostic algorithms. Using this gives the final result in Table 10. This shows that the cause of the alarm situation is that the steam valve does not transport enough steam. A failure value of 90 means that F9 and F12 are "very failed". A consequence value of 0.7 means that F9 is considered to be a "very primary" alarm. F12 is considered to be a "very secondary" alarm since the consequence value is –0.9. F13 is almost working as expected.

	Failure	Consequence
F9	90	0.7
F12	90	–0.9
F13	10	0.0

Table 10: *The result after defuzzication.*

4 Conclusions

MFM alarm analysis using fuzzy logic may solve some of the problems the already existing discrete logic algorithm would not handle, for example, chaotic switching due to noise and closeness to a decision limit. One possible drawback of the proposed alarm analysis algorithm is that it may be slower than the already existing algorithm, because of the increased computational complexity. However, since the already existing MFM algorithm is very fast, the fuzzy logic algorithm may still be among the fastest algorithms available.

5 Acknowledgements

I would like to thank my supervisor Jan Eric Larsson and my colleague Bengt Öhman at the Department of Information Technology for their support. I would also like to thank Anu Uus for her help with the proof-reading. This project is funded by TFR project no. 96–187.

References

1. Cox, E., Fuzzy Systems Handbook, Academic Press, London, 1994.
2. Larsson, J. E., Knowledge-Based Methods for Control Systems, Doctor's thesis, TFRT–1040, Department of Automatic Control, Lund Institute of Technology, Lund, 1992.
3. Larsson, J. E., "Diagnosis Based on Explicit Means-End Models," Artificial Intelligence, vol. 80, no. 1, pp. 29–93, 1996.
4. Larsson, J. E. and F. Dahlstrand, "New Algorithms for MFM Alarm Analysis", invited paper, Proceedings of the IEEE International Conference on System, Man, and Cybernetics, San Diego, California, 1998.
5. Larsson, J. E. and B. Hayes-Roth, "Guardian: An Intelligent Autonomous Agent for Medical Monitoring and Diagnosis," IEEE Intelligent Systems, vol. 13, no. 1, pp. 58–64, 1998.
6. Lind, M., "Representing Goals and Functions of Complex Systems-An Introduction to Multilevel Flow Modeling," Technical report, 90–D–38, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, 1990
7. Zadeh, L. A., "Fuzzy Sets," Information and Control, vol. 8, pp. 338–353, 1965.