Reliability, Risk and Safety – Ale, Papazoglou & Zio (eds)
© 2010 Taylor & Francis Group, London, ISBN 978-0-415-60427-7

Method for assuring that self-imposed changes made by adaptive systems do not compromise safety

André A. Hauge
OECD Halden Reactor Project, Halden, Norway
Department of Informatics, University of Oslo, Norway

Terje Sivertsen
OECD Halden Reactor Project, Halden, Norway

Ketil Stølen
Sintef ICT, Oslo, Norway
Department of Informatics, University of Oslo, Norway

ABSTRACT: This article presents a method consisting of 5 steps for assuring that self-imposed changes made by adaptive systems do not compromise safety. The method is intended to be used within the field of safety critical systems development with respect to applications where adaptive neural networks are part of some process control loop. The method is to provide a means for assuring before commissioning that all self-imposed changes resulting from the adaptivity after commissioning will have no adverse effects. This is achieved by: 1) identifying and specifying the potential hazards induced by the adaptiveness which must be handled during operation; 2) proposing a mechanism for separating the process of control and the process of adaptation such that only self-imposed changes with no effect on safety are effectuated; 3) providing a means for assessing online that self-imposed adaptive changes do not compromise safety.

1 INTRODUCTION

One of the basic activities when developing safety critical systems is the identification of operational and functional hazards. In order to obtain safety approval, all identified hazards must be addressed in such a way that documentable evidence that these are satisfactorily handled can be provided. The method proposed in this article is related to the assessment of a specific family of systems, namely adaptive neural networks. The knowledge/functionality of a neural network is embedded within its nodes and weights. In order to assess the functionality with respect to the presence of hazardous properties, potential hazards must first be identified. As the inherent functionality of an adaptive neural network changes during operation due to its adaptiveness, safe functionality may be replaced by unsafe functionality. We therefore need specialized methodology for assessing that self-imposed adaptive changes do not compromise safety. In this paper we outline a method supporting online assessment of the inherent functionality of the neural network against identified hazards and thus provide an approach to handle the negative effects associated with adaptiveness. This is achieved by assuring that no change during operation is effectuated before it has been assessed that there are no adverse effects. In order to provide this assurance, an online but out-of-the-loop adaptation and verification strategy is proposed as a mechanism to allow separation between the process of adapting the controller and the process of control; this is motivated and discussed in section 2.

The strategy provides a means for safe adaptive control by allowing refined versions of the control software replace the controller operating in-the-loop in an iterative adapt-verify-replace process. In order to be practically feasible, this approach requires that the processes related to the online handling of adaptation and verification can be effectively automated.

This article is primarily focused on verification issues, and then on the ability to verify that the control software satisfies safety requirements. In order to establish an effective verification process, our method emphasises hazard identification and a subsequent handling supported by formal methods in the following major steps:

1. Hazard identification and analysis
2. Risk handling

3. Formalisation of safety assertions
4. Knowledge acquisition on adaptation
5. Verification analysis Step 1) accommodates issues related to adaptivity and the ability to inflict self-imposed changes. In step 2) special attention is made to the risks associated with the adaptive feature. Step 3) describes the formalisation of safety requirements written in natural language, the result from step 2), into safety assertions in predicate logic. Step 4) describes the acquisition of knowledge related to change inflicted by the adaptation process; in this article this is acquisition of the inherent functionality of a neural network. Step 5) verifies that the functionality of the network fulfils the safety assertions.

This article is structured as follows: Section 1 provides a motivation for and introduction to our neural network assessment approach. Section 2 describes the challenges to be addressed. Section 3 defines a system used as an example throughout the article. Section 4 elaborates upon the process of identifying hazards in neural network applications. Section 5 addresses issues of risk management focusing on the need to mitigate some hazards during operation. Section 6 describes how safety requirements provided as a result from previous steps can be transformed into safety assertions. Section 7 gives a proposal for the acquisition of knowledge from a neural network. Section 8 describes a verification method, and section 9 concludes and presents further work.

2 SAFETY CHALLENGES TO BE ADDRESSED

Adaptive systems are proposed or explored in many different domains, e.g. health, nuclear power production, space exploration and military applications, for safety and mission critical control tasks as indicated in the papers (Beda et al. 2010), (Nesterov et al. 2009), (Sierhuis et al. 2003), (Soares et al. 2006) and (Tallant et al. 2006). The motivation is to increase performance by applying techniques providing dynamic behaviour capable of adapting some control function to handle unforeseen or otherwise uncertain factors. Evidently there are challenges with respect to determinism and predictability of adaptive systems since the behaviour of these types of systems will change over time as adaptations are effectuated. In a critical application, where there may be unacceptable consequence of certain failures, issues related to the possible negative consequence of adaptiveness must be resolved in order to allow utilisation. There are open questions with respect to how to assess the safety of such systems, (Soares et al. 2006) and (Tallant et al. 2006). (Tallant et al. 2006), addressing verification and validation needs in emerging safety critical control systems in the context of military aerospace applications identify non-determinism of intelligent and reasoning systems as what truly challenges current practices.

If an adaptive neural network is only allowed to operate in system states where there are no safety implications of failure, assessment becomes trivial but clearly reduces the applicability of the network. In order to achieve high applicability, an adaptive neural network must be allowed to operate in system states where there are safety implications of failure. In order to provide assurance that the neural network will not operate hazardously in these states there are basically two choices:

i. Provide assurance before commissioning that the adaptation algorithm cannot evolve the network into a hazardous configuration.
ii. Provide assurance before commissioning that no change during operation is effectuated before it has been assessed that there are no adverse effects.

The two choices are basically different in that choice i) establishes the safety argument once and for all before commissioning while choice ii) allows online assessment of change as an integral part of the safety argument.

With respect to online learning artificial neural networks for use in safety and mission critical context, requirements related to and means for verification and validation are addressed in (Kurd 2005) and (Taylor 2005). (Kurd 2005) describes the SCANN (Safety Critical Artificial Neural Network) model and how it can be applied in order to satisfy performance goals as well as provide safety argumentation. The SCANN incorporates constraints as part of the network model in order to constrain adaptation in the manner of choice i) above. Both (Kurd 2005) and Taylor (Taylor 2005) discuss several methods found in the literature addressing different aspects of the lifecycle important for verification and validation of a network. Related to acquisition of the knowledge represented by a trained neural network, the properties specifying the network function, both authors identify rule extraction techniques as a promising approach for demonstrating consistency with requirements. A survey of rule extraction techniques is provided in (Darbari 2000). (Darbari 2000) identifies that many of the approaches to rule extraction is in the form of a search problem where a space of candidate rules are explored and confirmed by testing of individual candidates against the network in order to establish valid rules. Although rule extraction techniques may offer a promising approach for assessing what a neural network has learnt, we will pursue a formal approach in this article, enabling the establishment of formal proofs with respect to properties of the system.

In order to achieve ii) we propose an online but out-of-the-loop adaptation and verification strategy. The main idea is to adapt and operate on separate instances of the control software. The instance which is allowed to be adapted operates out-of-the-loop in a background process. The instance operating in-the-loop at the sharp end is not allowed to adapt and behaves deterministically. The goal of the adaptation process is to refine the control software such that it provides a controller with as high performance as possible. Changes in the operating conditions may represent causes that trigger the need to adapt. Regardless of what triggered the need to adapt, once a refined version of the adapted system has reached some level of performance, the adaptation of this instance is stopped and the refined version undergoes verification with respect to the safety requirements. Once the refined version is successfully verified it may replace the controller operating in-the-loop. As the in-the-loop controller is updated, the adaptation process may start adapting its instance for a future release of a refined version of the controller. The process of adaptation, verification and update is intended to iterate in a loop providing an incremental refinement of the control software where each refinement replacing the in-the-loop controller is assured to satisfy the safety requirements.

In this article, related to the assessment and verification of an online learning neural network, we pursue choice ii) and the strategy of online but out-of-the-loop adaptation because it:

- Allows safety assurance by assessment of the product (the adapted network), not of the process (the adaptation algorithm). The positive effect is that only concrete network proposals are assessed. As part of the verification is automated and performed during operation, a challenge is to establish an effective online verification process.
- Supports minimal constraints on the neural network training algorithm or other mechanisms for adaptation since the safety argument does not rely on the adaptation process.
- Supports separation between how to achieve functional goals and assessment of consistency with safety constraints. If the adaptation algorithm produces an updated network that does not pass verification, the effect is reduced ability to meet performance goals, but it does not affect safety as the executing version of the network is already verified against safety requirements. In order to reduce the number of neural networks that fails verification, the knowledge acquired during verification should be communicated to the adaptation process.

3 EXAMPLE OF A SYSTEM

3.1 *The system*

The system in Figure 1 is taken from (Guarro et al. 1996), with some extensions and modifications, and will be used throughout this article. The operation of the system is basically the same as described in (Guarro et al. 1996) and defined as follows:

- A water tank is fed with water from an inflow pipe where a water pump is assumed to operate at constant speed supplying flow of water through the piping. The level of water in the tank is regulated by control- and stop valves on the inflow and outflow pipes. There are placed sensors to identify the water level, inflow and outflow.
- A tank bypass is allowed for emergency mode as tank overflows. In this mode, the inflow and outflow pipes are directly connected, and the tank is isolated via the actuation of the three stop valves located on the inlet and outlet sides of the tank piping.
- The tank flow and level control logic is implemented in an artificial neural network.

3.2 *Neural network*

The neural network in Figure 2 specifies input and output nodes which interface with the sensors and actuators to represent the component *Neural Network* in Figure 1. Different neural network experts might propose a variety of network implementations to solve the control problem at hand. The network in Figure 2 is just an example of a possible configuration illustrating a perceptron network, a network model originally developed by (Rosenblatt 1958), also known as a multilayer feed-forward network, with one hidden layer and full connection from one layer to the next. As the

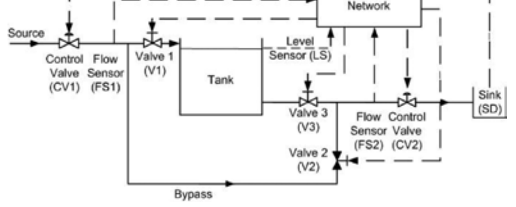

Figure 1. Example: Tank system.

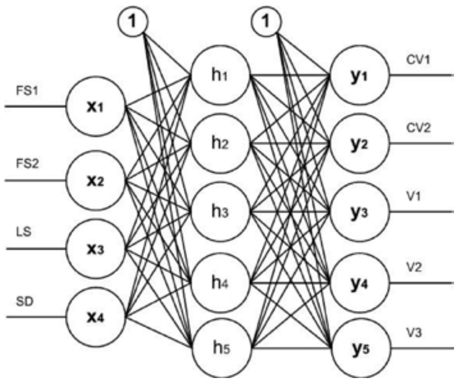

Figure 2. Example: Neural network.

network can be adapted online, the internals of the network will change depending on the constraints or lack of such in the adaptation algorithm. It is however expected that the input and output nodes will remain the same as these are the interfaces to the rest of the system.

A neural network like the one shown in Figure 2 may be adapted with e.g. a backpropagation training algorithm (Rumelhart et al. 1986) or other means in order to establish an effective neural network learning process. As stated in section 2, we want to pursue the assessment of the product, the network after training, rather than the adaptation process e.g. represented by a training algorithm. As a consequence we assume that there exists an effective process for the adaptation of our network such that performance goals are met. In the rest of this paper the focus is on describing a possible method for the assessment of safety issues and the verification of safety requirements.

4 STEP 1: HAZARD IDENTIFICATION AND ANALYSIS

4.1 *Preliminary Hazard Identification* (*PHI*)

In the PHI phase the goal is to identify the operational hazards related to the use of a system. Inputs are typically incident and accidents data, high-level functional requirements, system context descriptions and high-level design specifications. Results of the PHI are typically recorded in a preliminary hazard list and are used to re-evaluate the nature of the system or refine the system design in order to reduce hazards. The system is addressed at an abstract level, focusing on the interaction between the system and the environment in order to identify unwanted effects. HAZOP (IEC61882 2001) is one recognized method which supports the identification of hazards.

Assume that a systematic PHI has been conducted on the system in Figure 1 with a commonly accepted method and produced the following hazard list:

H1 Less water in Tank than the lower bound water level
H2 More water in Tank than the upper bound water level
H3 Less downstream flow to Sink than lower bound downstream flow
H4 More downstream flow to Sink than upper bound downstream flow

4.2 *Functional Hazard Analysis* (*FHA*)

FTA (Fault Tree Analysis) (IEC61025 2006) is one means to find the causes of hazards. The hazards from section 4.1 constitute the top events in a fault tree analysis where the causes of top events are systematically deduced in a top-down fashion. The goal of this analysis is to find the minimal cut sets. The minimal cut sets define the minimal sets of conditions that must be present in order for the top event to occur.

This article is not about increasing the robustness of a system in terms of tolerance to faulty components. The intention is to verify that the control logic embedded within a neural network is consistent with safety requirements. In doing so, all components of the system are assumed to behave as intended. The FTA (IEC61025 2006) serves as a means to establish the conditions, the states of the individual components in combinations that will lead to the top event.

Assume that a systematic FTA (IEC61025 2006) has been conducted on the system in Figure 1 taking the hazards from section 4.1 as top events and produced fault trees like in Table 1 where the hazard H3 is analysed. The fault tree in Table 1 specifies that the hazard H3 can be caused by the following three sets of conditions, constituting the minimal cut sets for the hazard:

C1 Control valve CV2 opening is below level lower bound
C2 The combined event that both the stop valve V2 and V3 are closed at the same time
C3 The combined events that stop valve V2 is open and V3 is closed (bypass mode) and the control valve CV1 opening is below lower bound

The fault tree analysis is a means to identify safety requirements. Based on the evaluation of the cut sets, the safety requirements as defined in Table 2 are specified as a means to prevent the events associated with the hazard H3 as described by the fault tree in Table 1.

Table 1. Fault tree.

FTA-H3: Less downstream flow to Sink than lower bound downstream flow					
	OR				
Downstream flow regulating valve set to less than lower bound	No or less flow through Tank or Bypass				
CV2 less than lower bound	OR				
	No flow through Tank or Bypass		Less flow through Tank or Bypass		
	AND		AND		
	V2 Closed	V3 Closed	V2 Open	V3 Closed	CV1 Less than lower bound

Table 2. Example safety requirement.

ID	Requirement
SR-H3-C1	Control valve CV2 opening shall always be greater than or equal to lower bound
SR-H3-C2	Stop valves V2 and V3 shall never be closed at the same time
SR-H3-C3	If stop valve V2 is open and V3 is closed then control valve CV1 shall always be grater than or equal to lower bound

5 STEP 2: RISK HANDLING

Given that the hazard identification and analysis phase has been conducted. The subsequent risk handling phase typically evaluates each hazard with respect to likelihood of occurrence to provide a notion of risk. Further, identified risks are then evaluated against some predefined notion of acceptable and unacceptable risk. A hazard associated with some measure of likelihood of occurrence constitutes a risk which can be handled in basically two ways:

i. Reduce the risk to an acceptable level by some means of avoidance or design mitigation.
ii. Accept the risk and leave it unaddressed as it is at an acceptable level.

We evade the issue of risk handling under the assumption that the likelihoods associated with the hazards defined in section 4.1, related to the system in Figure 1, are acceptable given that the control logic embedded within the neural network can be guaranteed to satisfy the safety requirements specified in Table 2. This means that although the system is susceptible to hardware component failures, the individual component failure rates are within an acceptable level giving sufficient level of safety as long as the software driven control logic satisfies applicable safety requirements.

6 STEP 3: FORMALISATION OF ASSERTIONS

In this third step of the method, it is assumed that the preceding steps of the method results in a complete and correct set of safety requirements. This means that in order to verify that the system is safe, we can verify that the neural network controls the actuators of the system in accordance with safety requirements like those exemplified in Table 2. The safety requirements specify mandatory behaviour and address the identified hazards in such a way that they prevent unwanted events. In order to demonstrate that the system is safe, our method emphasizes demonstrating that the control logic embedded within the neural network structure and parameters satisfy the safety requirements with respect to functional behaviour.

The software part of a system fails systematically. Given a set of well defined functional safety requirements, and given well defined software, it is possible to formally argue that the software either satisfies the requirements or conversely does not. In order to provide a compelling argumentation that the neural network software satisfies the requirements, our method proposes a formal reasoning approach.

The starting point of the formalisation process are the safety requirements as in Table 2. The results of the formalisation are referred to as safety assertions. The safety assertions are specified in

Table 3. Safety assertions.

ID	Assertion
A-SR-H3-C1	$\forall s$: *SYSTEM*; *cvOpEqMoreThan*(*cv*2, *lb*, *s*)
A-SR-H3-C2	$\forall s$: *SYSTEM*; $\neg$(*isClosed*(*v*2, *s*) $\wedge$ *isClosed*(*v*3, *s*))
A-SR-H3-C3	$\forall s$: *SYSTEM*; (*isOpen*(*v*2, *s*) $\wedge$ *isClosed*(*v*3, *s*) $\Rightarrow$ *cvOpEqMoreThan*(*cv*1, *lb*, *s*)

Table 4. Types, constants and functions.

Types	
SYSTEM	Set of states representing the neural network software system
CONTROLVALVE	*cv1, cv2*
STOPVALVE	*v2, v3*
Constants	
BOUND	*lb*
Functions	
isClosed	*STOPVALVE* $\times$ *SYSTEM* $\rightarrow$ *BOOL*
isOpen	*STOPVALVE* $\times$ *SYSTEM* $\rightarrow$ *BOOL*
cvOpEqMoreThan	*CONTROLVALVE* $\times$ *REAL* $\times$ *SYSTEM* $\rightarrow$ *BOOL*

first order predicatelogic as indicated in Table 3 and Table 4. Each row in Table 3 is a formalisation of the corresponding row in Table 2. The safety assertions provide a means to establish an effective process for assessment of the inherent functionality of a neural network with respect to the safety requirements, which will be described in step 4 and 5, section 7 and 8 respectively.

7 STEP 4: KNOWLEDGE ACQUSITION ON ADAPTATION

In section 2 we advocate the principle of online but out-of-the-loop, iterative adaptation and subsequent verification process. This principle allows adaptation in increments where the adaptation and verification process is separated from the process executing the neural network function. As training is stopped and there will be no changes to the neural network in-the-loop, the internal properties of the network at a specific adaptation increment will not change and thus is static and provide a deterministic function over the input. Acquisition of knowledge related to a refined version of a neural network can easily be obtained by extracting the internal properties defining the network from its specification, e.g. the values of the weights and biases.

A neural network represents a mathematical function. The network in Figure 2 fulfils its function f: $X \rightarrow Y$ by a combination of simple arithmetic and transformational operations. The following properties are assumed to be contained in the neural network specification:

- The domain of the input nodes which specify the possible input activation.
- The range of the output nodes which specify the possible output activation.
- Properties defining the interconnection between the nodes like arcs, and weights.
- Properties defining the node function e.g. how inputs are handled, any bias term and activation function.

The network in Figure 2 may be represented as a series of assignments where the activation of a node (not input node) is specified as an expression over the nodes in the preceding layer. If we assume for simplicity that the input nodes simply pass their values forward to the hidden layer, the domain and range for each input node are identical. The domain of the hidden nodes and the output nodes can be deduced from the range of the nodes in the preceding layer and the properties specifying the interconnection to these nodes like arcs, weights and bias. The range of the hidden nodes and the output nodes can be deduced from knowledge concerning their domain and transfer function. Once the specification is established it is possible to formally deduce important properties of the system by the use of substitution and predicate transformation techniques which is the focus of step 5 described in section 8.

Before we look into the verification issues with respect to a given specification, an example of a specification is provided. The network in Figure 2 could be represented by the assignments (1), (2) and (3) which provide a definition of a multilayer perceptron network. The network is a function f: $X \rightarrow Y$ such that:

$$f(x_1, \ldots, x_n) = y_1, \ldots, y_p \tag{1}$$

$$h_m = g\left(\sum_{i=1}^{n} x_i w_{im} + \beta_{1m}\right);\ g(x) = 1/1 + e^{-x} \tag{2}$$

$$y_p = g\left(\sum_{i=1}^{m} h_i z_{ip} + \beta_{2p}\right);\ g(x) = 1/1 + e^{-x} \tag{3}$$

The assignment (2) specifies that a hidden node h_m is assigned a value by a sigmoid transfer $g(x)$ of the sum of the weighted input nodes values plus a bias β. Likewise, assignment (3) specifies that an output node y_p is assigned a value by a sigmoid transfer $g(x)$ of the sum of the weighted input nodes values plus a bias β. Once values are assigned to the variables $x_1, \dots, x_n$ the function propagates the values forward in the structure by successive assignments in order to obtain the values on $y_1, \dots, y_p$. It is not presented here how the values of the properties w_{im}, z_{ip} and β are obtained. Given that supervised training can be applied then the backpropagation training algorithm (Rumelhart et al. 1986), is a commonly used method. The training algorithm adjusts the internal properties of the network such that it satisfies the intended function to some acceptable level of error. From section 2, we have that the out-of-the-loop adaption and verification strategy supports minimal constraints on the neural network training algorithm since the safety argument does not rely on this process. We therefore can pursue verification of the trained network and not the training process. In the next step of the method, described in section 8, it is assumed that training is stopped such that the candidate version of the network contains concrete values for the weights and biases.

8 STEP 5: VERIFICATION ANALYSIS

The neural network can be seen as a function taking several parameters and providing several outputs in one function call, accomplished by a series of successive assignments. The verification analysis could therefore be based on the weakest precondition calculus by (Dijkstra 1975). The weakest precondition calculus states that for a given statement S and a postcondition Q, wp(S, Q) denotes the the weakest precondition of S satisfying Q.

If we assume that the neural network specification given in the assignments in section 7 represents S and that the safety assertion S3-H3-C1 in Table 3 represents Q then it can be shown that S will fulfil Q if any combination of input provided by the sensors connected to the neural network fulfills wp(S, Q). Provided a similar deductive argumentation for all relevant safety assertions, then our system can be regarded safe. As an example of the wp calculation, assume Figure 3 represents the neural network after adaptation. If we assume, unrealistically but for the sake of making a simple example, that the β's in this case are all zero, the hidden nodes simply pass the weighted sum of their inputs forward, and the output nodes uses a step function function then (4), (5) and (6) specifies the relevant part of a function S to be analysed. Assertion A-SR-H3-C2, specifying the predicate $\neg(isClosed(v2, s) \wedge isClosed(v3, s))$, can be rewritten in terms of the nodes of the network as $y_4 = 1 \vee y_5 = 1$.

$$f(x_1, \dots, x_4) = y_4, y_5 \qquad (4)$$

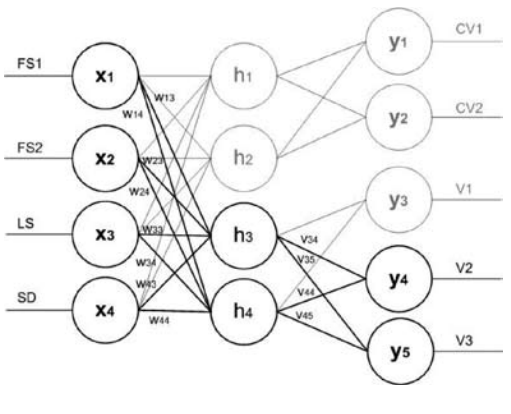

Figure 3. Reduced neural network.

$$h_3 := x_1{*}w_{13} + x_2{*}w_{23} + x_3{*}w_{33} + x_4{*}w_{43};$$
$$h_4 := x_1{*}w_{14} + x_2{*}w_{24} + x_3{*}w_{34} + x_4{*}w_{44}; \qquad (5)$$

$$if(h_3{*}v_{34} + h_4{*}v_{44} > \theta_{y4})\ then\ [y_4 := 1]\ else\ [y_4 := 0];$$
$$if(h_3{*}v_{34} + h_4{*}v_{45} > \theta_{y5})\ then\ [y_5 := 1],\ else\ [y_5 := 0]; (6)$$

The wp calculation on S gives the result provided in (7):

$$\begin{aligned} &wp(S, y_4 = 1 \vee y_5 = 1)\\ &\quad = ((x_1{*}w_{13} + x_2{*}w_{23} + x_3{*}w_{33} + x_4{*}w_{43}){*}v_{34}\\ &\quad + (x_1{*}w_{14} + x_2{*}w_{24} + x_3{*}w_{34} + x_4{*}w_{44}){*}\ v_{44} > \theta_{y4}) \vee\\ &((x_1{*}w_{13} + x_2{*}w_{23} + x_3{*}w_{33} + x_4{*}w_{43}){*}v_{35}\\ &\quad + (x_1{*}w_{14} + x_2{*}w_{24} + x_3{*}w_{34} + x_4{*}w_{44}){*}\ v_{45} > \theta_{y5}) \end{aligned} \qquad (7)$$

After a period of training, the values of the weights w and v, threshold values θ, and bias values β are known. In addition, the domain of the input nodes are known. To derive the preconditions, or in other words the activation patterns, that satisfy the postcondition it is straightforward to insert these values in the expression provided in 7 to finalise the calculation.

In order to establish an effective verification process with the weakest precondition calculus method, problems related to combinatorial explosion needs to be addressed. As the number of input nodes increase, and possibly also may take real values in some interval, infinitely many activation patterns might establish a postcondition.

9 CONCLUSIONS

In this paper we advocate a strategy of online, out-of-the-loop, iterative adaptation and verification in order to separate the process of adapting and the process of control. This strategy requires a mechanism disallowing the in-the-loop control software to be replaced by a refined version before the refined version has been successfully verified against safety requirements. Given such a mechanism, and the possibility to effectively adapt and verify online, we offer possibilities for safe adaptive control. The 5 step method outlined in this article support pre-commission safety assurance argumentation for systems utilising adaptive components by the use of risk assessment techniques to identify functional safety requirements, and formal methods for argumentation related to a systems ability to satisfy these requirements. The set of functional safety requirements represents a specification of behaviour such that if it is satisfied, establishes that the function is safe. The method is not concerned with the set of functional requirements which have no impact on the safety property of the system. A violation of the purely functional requirements affects performance rather than safety and it is assumed that performance issues are addressed more effectively by other means than contained in this proposal. The main hypotheses argued by this article are:

i. Possible negative effects of adaptiveness can be mitigated by a mechanism separating the process of adapting and the process of control such that a change is not effectuated before its consistency with requirements has been verified.
ii. Hazard identification and analysis techniques are suitable to identify operational hazards with respect to the possibly unpredictable behaviour of adaptive neural networks and provide a basis for defining safety requirements.
iii. Given a set of valid safety requirements specified as formal expressions, the possibly negative impact on safety by self-imposed changes can be assessed online through processes analysing the behaviour of the adapted system against the requirements.

The method outlined in this article is a proposal for how to handle potential negative effects of adaptiveness in high integrity systems. Related work by (Kurd 2005) approaches the problem with a different strategy than pursued in this article. (Kurd 2005) describes the SCANN model which apply constraints in the network such that it cannot evolve into a hazardous configuration. Safety assurance claims related to application of the SCANN model are all based on pre-commission arguments. The strategy pursued and methods emphasised in this article are related to the work presented in (Mili et al. 2004) and (Liu et al. 2007). (Mili et al. 2004) describe a framework for verification of online learning systems. The framework presents tentative workon formal methods and refinement calculi techniques as means to address issues of safe learning. Particularly emphasised in the article is the importance of training data on the system behaviour. Some of the same authors contribute in (Liu et al. 2004) describing an online verification strategy like pursued in this article, then by the use of monitors. The article presents two techniques, one for detecting novel inputs following the argument of the training data importance on behaviour, and a second technique used to monitor the control outputs. The techniques are used primarily to address performance issues by detecting anomalies in the learning data and to provide a reliability measure of the network after a change has been effectuated. The paper does not clearly state how to establish if it is safe to effectuate the change.

Further work on the method proposed in this article includes a refinement and detailing of the method and evaluation of its feasibility in case studies. Particularly interesting with respect to feasibility studies of the method is indication on scalability. One of the challenges of the method is possibilities for combinatorial explosion as e.g. the number of nodes grows beyond what exemplified in this article.

REFERENCES

Beda, A., P. Spieth, T. Handzsuj, P. Pelosi, N. Carvalho, E. Koch, T. Koch, and M. Gama de Abreu (2010). A novel adaptive control system for noisy pressure-controlled ventilation: a numerical simulation and bench test study. *Intensive Care Med 36*(1).

Darbari, A. (2000). Rule extraction from trained ANN: A survey. Technical Report WV-2000-03, Department of Computer Science, Dresden University of Technology, Germany.

Dijkstra, E. (1975). Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM 18*(8), 453–457.

Guarro, S., M. Yau, and M. Motamed (1996). Development of tools for safety analysis of control software in advanced reactors. Technical Report NUREG/CR–6465, Nuclear Regulatory Commission, Washington DC, United States.

IEC61025 (2006). *IEC 61025: Fault tree analysis (FTA)*. International Electrotechnical Commission.

IEC61882 (2001). *IEC 61882: Hazard and operability studies (HAZOP studies) - Application guide*. International Electrotechnical Commission.

Kurd, Z. (2005). *Artificial Neural Networks in Safety-critical Applications*. Ph. D. thesis, Department of Computer Science, University of York.

Liu, Y., B. Cukic, and S. Gururajan (2007). Validating neural network-based online adaptive systems: a case study. *Software Quality Journal 15*(3), 309–326.

Mili, A., G. Jiang, B. Cukic, Y. Liu, and R. Ben Ayed (2004). Towards the verification and validation of online learning systems: General framework and applications. *Hawaii International Conference on System Sciences 9*, 90304.1.

Nesterov, Y., M. Pikin, and E. Romanovskaya (2009). Development of technological algorithms for automated process control systems of power units. *Thermal Engineering 56*(10), 827–831.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review 65*(6), 386–408.

Rumelhart, D., G. Hinton, and R. Williams (1986). Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: foundations*, 318–362.

Sierhuis, M., J. M. Bradshaw, A. Acquisti, R. van Hoof, R. Jeffers, and A. Uszok (2003). Human-agent teamwork and adjustable autonomy in practice. In *Proceeding of the Seventh International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Soares, F., J. Burken, and T. Marwala (2006). Neural network applications in advanced aircraft flight control system, a hybrid system, a flight test demonstration. In *Neural Information Processing*, Volume 4234 of *Lecture Notes in Computer Science*, pp. 684–691. Springer.

Tallant, G., P. Bose, J. Buffington, V. Crum, R. Hull, T. Johnson, B. Krogh, and R. Prasanth (2006). Validation & verification of intelligent and adaptive control systems. In *IEEE Aerospace Conference*.

Taylor, B.J. (2005). *Methods and Procedures for the Verification and Validation of Artificial Neural Networks*. Springer.