proceedings

MDPI

Proceedings

Diagnostics and Prognostics of Energy Conversion Processes via Knowledge-Based Systems †

Roberto Melli and Enrico Sciubba *

Mechanical Engineering Department, University of Roma 1,00185 Rome, Italy; banshee700@yahoo.com

* Correspondence: enrico.sciubba@uniroma1.it

† Presented at the First World Energies Forum, 14 September–5 October 2020; Available online: https://wef.sciforum.net/.

Published: 14 September 2020

Abstract: This paper presents a critical and analytical description of an ongoing research program aimed at the implementation of an expert system capable of monitoring, through an Intelligent Health Control procedure, the instantaneous performance of a cogeneration plant. The expert system is implemented in the CLIPS environment and is denominated PROMISA as the acronym for PROgnostic Module for Intelligent System Analysis, generates, in real time and in a form directly useful to the plant manager, information on the existence and severity of faults, forecasts on the future time history of both detected and likely faults, and suggestions on how to control the problem. The expert procedure, working where and if necessary with the support of a process simulator, derives from the available real-time data a list of selected performance indicators for each plant component. For a set of faults, pre-defined with the help of the plant operator (Domain Expert), proper rules are defined in order to establish whether the component is working correctly; in several instances, since one single failure (symptom) can originate from more than one fault (cause), complex sets of rules expressing the combination of multiple indices have been introduced in the knowledge base as well. Creeping faults are detected by analyzing the trend of the variation of an indicator over a pre-assigned interval of time. Whenever the value of this ''discrete time derivative'' becomes ''high'' with respect to a specified limit value, a ''latent creeping fault'' condition is prognosed. The expert system architecture is based on an object-oriented paradigm. The knowledge base (facts and rules) is clustered: the chunks of knowledge pertain to individual components. A graphic user interface (GUI) allows the user to interrogate PROMISA about its rules, procedures, classes and objects, and about its inference path. The paper also presents the results of some simulation tests.

Keywords: energy systems; expert systems; systems prognostics

1. Introduction

Modern energy conversion plants are very complex systems under a technological point of view. Any downtime or drop in the energy quality of the output involve often unacceptable elevate direct and indirect monetary losses, but even more important is the resource destruction that constitutes the end result of the fault. All modern design methods contain procedures that take into due account variable load conditions (off-design operation), availability losses due to scheduled and unscheduled maintenance and performance degradation due to wear and fouling in the equipment.

To lessen the likelihood of plant failures, preventive maintenance is regularly performed: it reduces by three to nine times the costs in lost production [1], higher costs for parts, and other overhaul costs compared to reactive, unplanned maintenance.

Among the different types of preventive maintenance, Condition-based Maintenance (CBM) presents several advantages when applied to energy conversion systems. CBM is a maintenance

strategy that monitors the actual health evolution in time of the plant operational performance and reports the cause(s) of the detected malfunctioning. Therefore, maintenance will only be performed when certain indicators show signs of decreasing performance or upcoming failure. These indicators include non-invasive measurements, performance data and scheduled tests.

Compared with preventive maintenance, CBM thus increases the time between maintenance interventions, because maintenance is done on an as-needed basis [2–4].

Condition data can be gathered either at certain intervals, or continuously. The analysis of the data flowing from the plant becomes rapidly overwhelming, which make it difficult to analyze. A powerful aid to this task is provided by the implementation of an expert system. The choice of a knowledge-based expert system rather than a deep learning solution is suggested because a lack of training data makes machine learning approaches fall short. Moreover, the expert system works toward explainable AI and expands the knowledge through collaborative interactions. Most modern AI algorithms are like black boxes resulting in answers and recommendations without any insight into how the system arrived at those answers and which parameters were most significant.

Intelligent process management tools (IPMTs) [5,6] are not only by definition capable of producing an intelligent diagnosis of the present state of the plant but also to enact a prognostic action, making intelligent estimates of the future state of the plant under the foreseen boundary conditions [7]. Finally, they can use design, operation and load-scheduling data, together with other relevant external information (like for instance local weather forecasts or projected operating load curves of similar plants in the same ''fleet'') to provide operators with valuable information about the ''optimal'' operating curve of the plant in some future period T [8].

The present paper describes the development of a diagnostic and prognostic tool, specifically designed for a gas turbine-based cogeneration system; its development constitutes though a useful paradigm for different applications.

Let us define the plant availability factor PF as the ratio of the total equivalent full load operating hours in a year and the total number of hours in the year. It is apparent that no energy conversion plant can operate with PF equal to 1, due to three orders of reasons:

(a) plant shutdowns due to scheduled maintenance;
(b) plant shutdowns due to unscheduled maintenance;
(c) plant shutdowns due to sudden failures.

It is useful for our purposes to separately account for events of type ''b'', that imply the replacement of a component for which an early failure has been prognosed, and events of type ''c'', in which the replacement is done after the failure has forced a plant shutdown.

Our study specifically concentrates on plant shutdowns due to sudden failures. Strictly speaking, ''sudden catastrophic failures'' rarely happen as such, and when they do, they are obviously by definition unforeseeable. But extensive field studies have conclusively shown that most of the failures we call ''sudden'' are in reality caused by a series of component-localised phenomena that lead to a (usually very small but still significant) deterioration of its performance.

Our efforts may thus be redirected to the early detection of these ''performance degradation''-warning signals. The method to follow is in principle straightforward: a sufficient number of ''critical points'' in the process are monitored in real time, and a specific series of performance decay indicators are computed. As soon as one of these faults is detected, the operator, working under tight co-operation with the designer and the plant manager decides whether to execute an immediate shutdown to fix the fault, or to wait until the next scheduled maintenance intervention.

2. The General Conceptual Layout of a Diagnostic/Prognostic System

In the language of artificial intelligence (AI), we say that a procedure is enacted by an ''Agent''. In the following description, the agent is our expert system: but it is easy to recognise a high degree of similarity between the individual steps of the procedure and the actions that a human operator would take when executing the same task. Our scope here is to show that both the procedures in its entirety and each one of its single steps is feasible at the present level of AI technology. We shall

separately describe the diagnostic and the prognostic procedures, but will show later that they both admit a meta-procedure, i.e., they can be embedded in a single code.

2.1. A Diagnostic System

A possible procedure for an automatic diagnostic system consists of the following steps:

1. The intelligent agent (IA) must identify in "real time" the operational state of the process. This requires that the IA be endowed with an efficient interface with a process data collection system which produces vector of length N containing an ordered set of measurables, i.e., of process parameters that identify the state (mass flow rates, pressures, temperatures, etc.).
2. At each selected time step, the IA must compare, at each selected time step, the detected operational state with the expected one. To do this, IA must have access either to a pre-determined operational process schedule of the process, or, if the latter is not available, to a reliable process simulator that provides the IA with such reference operating state.
3. If the value of the kth measurable differs from the corresponding design value by more than a preset tolerance, the IA activates a monitoring-and-control procedure on the component to which this measurable pertains.
4. The IA verifies whether the ''failure'' condition just detected appears in one of the ''fault chains'' contained in its knowledge base. If it does, then the IA proceeds to step 5 here below. If it does not, the IA activates a sub-procedure to monitor k for a prescribed period of time, and notifies the (human) plant operator of this action.
5. If the event ''kth measurable out of range'' belongs to one or more fault chains known to the IA, the agent launches a monitoring-and-control procedure on all measurables i, j, …, p that appear together with k in the detected fault chains.
6. If a fault chain is indeed identified as ''active'', the IA will: a—notify the plant operator; b—consult its knowledge base to search for remedial actions (e.g., adjustment of other process parameters to compensate for the derangement in k); c—decide whether it is possible to wait for the next scheduled maintenance intervention or a repair/substitution is immediately necessary.

2.2. A Prognostic System

A possible procedure for an automatic prognostic system consists of the following steps:

1. The IA must compare at a pre-determined time step the operational state of the process.
2. The IA projects the detected operational state forward in time, founding this projection on the most recent time history (two or more previous time steps) of the process.
3. If the projected value of the kth measurable at $t + \Delta t$ activates one of the known fault signatures, or if it shows an undesirable trend in the time history of x_k (e.g., "dx_k/dt too high" according to some norm), the IA activates a monitoring-and-control procedure on the component to which this measurable pertains.
4. The IA also launches a monitoring-and-control procedure on all measurables r, s, …, z that are related to k (i.e., whose values are known to be functionally linked to the value of x_k).
5. Otherwise, the IA keeps monitoring x_k for a pre-defined time interval, and notifies the plant operator of this action.
6. If the IA estimates that a fault chain may be ''activated'' by an excessive variation of x_k, it will:

   A. notify the plant operator;
   B. consult its knowledge base to search for and recommend actions (e.g., adjustment of other process parameters to compensate for the derangement in x_k
   C. decide whether it is possible to wait for the next scheduled maintenance intervention or a repair/substitution is immediately necessary.

Notice the remarkable analogy between the steps of the diagnostic and those of the prognostic procedure.

3. Theoretical and Practical Aspects of the Implementation of the Intelligent Agent

For the intelligent agent to be in fact ''expert'' and ''intelligent'' in performing his task, its knowledge base (KB) must be as ''complete'' and ''exact'' as possible:

- ''Complete'' means that there must exist a one-to-one mapping of all rules and information available to the human operator and this KB.
- ''Exact'' means that this mapping must be logically consistent, i.e., that no logical chain of induction correctly derived from the KB contradicts any of the rules and information available to the human operator.

3.1. The Meta-Rules of Failure Detection

It is known from AI theory [9,10] that it is convenient to re-organise, wherever possible, the knowledge bits acquired during the knowledge acquisition phase. Such a systematization goes in favour of the transparency and the accessibility of the ''built-in-logic'' of the expert system. In the case in point, we are dealing with ''failures'' of a system, and we have found it useful to construct our KB on the basis of the following seven meta-rules:

1. There exists a finite number of possible types of failure, and for each one of them there exists at least one specific signature, i.e., a unique combination of the process parameters.
2. There are no sudden failures: every possible failure is ''forewarned'' by a drifting of the point representative of the operational state of the plant, on a path that leads to a specific attractor in the state space (the failure point).
3. Each one of these ''drifting'' processes has a characteristic time scale that depends both on the component and on the type of failure.
4. A convenient way to represent such a drifting is that of employing a proper set of dimensionless indicators, each defined as the ratio of the instantaneous value of a measurable of interest to its ''design'' value. Notice that such a design value is in reality a time-dependent quantity: it is the value expected for the same instantaneous operative conditions but without any derangement.
5. The process of ''failure formation'' is described by at least one ''fault chain'', i.e., an ordered list of the immediate causes of the failure. There may be more than one chain (see point 6 here below). Each chain though has at least two fuzzy aspects: first, the ''causes'' it contains are necessary, but not sufficient (for example, for a creep failure in a first row statoric blade in a gas turbine, it is necessary that the gas temperature at turbine inlet be higher than a certain design limit; but once the temperature exceeds this limit, failures are not certain). Second, even this necessity is affected by some degree of uncertainty (for example, a blade failure may happen even if the gas temperatures are below the design limit).
6. Some of the fault chains may be concurrent. That is, the same failure stem from one or the other or from a combination of two (or more) fault chains.
7. Many of the fault signatures are non-local: the values of measurables detected at locations physically remote from the point where the failure actually takes place may be affected by the drifting process mentioned in point (2). In this case, we say that these measurables (and the indicators constructed on them) are correlated with the ones immediately affected by the failure.

3.2. Formalisation of the Fault Signatures and Choice of the Fault Indicators

A very extensive database is available for the monitoring of all energy conversion plants (e.g., ''efficiency'', ''mechanical output'', ''thermal output''), and especially for gas turbines and their derivates (combined and cogeneration plants), a very extended database is available. There is a body of international industrial standards, often validated by Public Agencies, which regulates even the fine details of the type and tolerance of the measurables. Our approach here is though rather different: we are not interested in the abidance by contractual specifications, but rather in a (continuous) monitoring of whether the system operates within a certain number of admissible states. Therefore, the various sets of measurables defined by International Standards do not suffice for our purpose: in

fact, our KB complements them with other knowledge bits derived from design handbooks, operators manuals and interviews with field experts. Using as an example a standard gas turbine plant, Table 1 reports a list of failures to be diagnosed/prognosed and Table 2 the indicators adopted.

Table 1. Examples of possible faults for a gas turbine.

Component	Possible Fault(s)	Component	Possible Fault(s)
Filter	Leakage Fouling	Secondary heat exchanger	Fouling
Compressor	Stall Choking Fouling Excessive exit temperature Malfunctioning	Primary loop	Fouling
Primary Combustion Chamber	Fouling Excessive pressure losses CH_4- or H_2O valve failure	Secondary loop	Fouling
Primary fuel Injector	Fouling Failure	Main pump	Cavitation Malfunctioning
Boiler stack	Secondary combustion reactions Fouling Leakage	Main shaft	Near-critical vibration frequencies
Turbine	Fouling Choking Excessive inlet temperature	Afterburner	CH_4 injector fouling
Primary heat exchanger	Fouling		

Table 2. Example of indicators.

Air filter	$I_1 = \Delta p/\Delta p_d$	Turbine	$I_{11} = \eta_c/\eta_{cd}$ $I1_2 = T_{ex}/T_{exd}$ $I_{13} = m_{ex}/m_{exd}$
Compressor	$I_2 = c_p\ \Delta T/c_p\Delta T_d$ $I_3 = \eta_c/\eta_{cd}$ $I_4 = \beta_c/\beta_{cd} = p_c/p_{cd}$ $I_5 = m_c/m_{cd}$	Electrical generator	$I_{14} = \omega_c/\omega_{cd}$

Combustion chamber	$I_6 = \Delta p_{cc}/\Delta p_{ccd}$ $I_7 = m_{cc}/m_{ccd}$	Afterburner	$I_{14} = \Delta T/\Delta T_d$ $I_{15} = m/m_d$
Fuel injector	$I_8 = m_{fi}/m_{fid}$	Main pump	$I_{16} = m_p/m_{pd}$ $I_{17} = \Delta p_p/\Delta p_{pd}$
Boiler main stack	$I_9 = X_{NOX}/X_{NOXd}$ $I_{10} = X_{CO2}/X_{CO2d}$	Shaft (vibrations)	$I_{18} = rms/rms_d$

3.3.1. The Mathematical Formulation

Define the ''performance function'' $\mathbf{\Pi_P}$ of an energy conversion process **P** as the deterministic mathematical relation between the instantaneous process output(s) and a set of N process parameters that we call the measurables: **ΠP** may be thought of as an operator that, applied to the vector **X** of measurables, generates the output vector **Y**, $\mathbf{\Pi_P(X) = Y}$. Now, denote as **X′** a deranged operational state, in which some of the measurables have taken values slightly, but detectably, different from the design values. The new functional value assumed by the operator $\mathbf{\Pi_P(X)}$:

$$\mathbf{\Pi_P(X')} \sim \mathbf{\Pi_P(X)} + d\mathbf{\Pi}/d\mathbf{X}\,(\mathbf{X} - \mathbf{X'}) \quad (1)$$

where $d\mathbf{\Pi}/d\mathbf{X}$ represents the term-by-term derivative of a vector and not the total differential and the new vector of measurable is:

$$\mathbf{\Pi_P(X')} = \mathbf{Y'} \quad (2)$$

3.3.2. The Knowledge Base Implementation

To complete the transition from the mathematical representation of component failure to the knowledge base construction it is necessary to create the so-called fault chains diagrams.

The first step is to define the ''performance indicators'' on which the derangement from ''standard operative conditions'' is measured. The plant operating manual and the design specifications provided by the designer and by the constructor define only a very limited set of operating points. We must add some form of ''logical extrapolation'' based on an intelligent comparison between the measured data and a set of proper theoretical operating curves (this step requires the assistance of a Domain Expert). At this point, we can create the fault chains for the knowledge base of the expert system.

An example of failure detection criteria for compressors are presented in Table 3.

Table 3. Failure detection criteria for compressors.

Compressor	Fouling	$I_2 = c_p\,\Delta T/c_p\Delta T_d$	(I_3 low U I_4 low) or (I_3 low U I_2 high)
	Malfunctioning	$I_3 = \eta_c/\eta_{cd}$	(I_3 high U W_c high)
	Choking	$I_4 = \beta_c/\beta_{cd} = p_c/p_{cd}$	(I_4 low U I_5 high) or (I_4 low U T_3 low)
	Excessive T_3	$I_5 = m_c/m_{cd}$	I_2 high
	Stall		I_4 low U I_5 low

The corresponding failure chain for compressor chocking is presented in Figure 1.

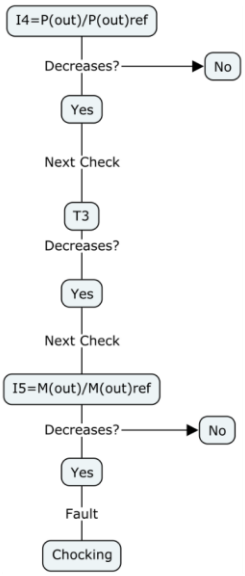

Figure 1. Compressor chocking fault chain.

The expert system, knowledge base and inference engine, have been written in CLIPS, an open-source programming environment. The inputs are fed to the knowledge base from a pre-processor that receives and elaborates data either from a data acquisition system or from a plant simulation. The code scans all fault indicators, establishes the progression at assigned time steps and feeds the results to the inference engine of the expert system. That consults the knowledge base to determine if there is a incipient fault and its possible causes.

In Figure 2 is presented an example of the code for the compressor fault chain.

```
 CLIPS (6.31 6/12/19)
CLIPS> (load "wef1 test.clp")
Defining deftemplate: rule
Defining defrule: propagate-goal +j+j+j
Defining defrule: goal-satified =j+j+j+j
Defining defrule: remove-rule-no-match +j+j+j
Defining defrule: modify-rule-match =j+j+j
Defining defrule: rule-satisfied =j+j+j
Defining defrule: ask-question-no-legalvalues
+j+j+j+j
Defining defrule: ask-question-legalvalues +j+j+j+j
Defining deffacts: knowledge-base
TRUE

CLIPS> (reset)
CLIPS> (run)
        Is the indicator I4 increasing? (yes no) yes
        Is the indicator I5 decreasing? (yes no) no
        All indicators are OK

CLIPS> (reset)
CLIPS> (run)
        Is the indicator I4 increasing? (yes no) yes
        Is the indicator I5 decreasing? (yes no) yes
        I think the failure is compressor_stall
```

```
CLIPS> (clear)
CLIPS> (load "wef1 test.clp")
Defining deftemplate: rule
Defining defrule: propagate-goal +j+j+j
Defining defrule: goal-satified =j+j+j+j
Defining defrule: remove-rule-no-match +j+j+j
Defining defrule: modify-rule-match =j+j+j
Defining defrule: rule-satisfied =j+j+j
Defining defrule: ask-question-no-legalvalues
+j+j+j+j
Defining defrule: ask-question-legalvalues +j+j+j+j
Defining deffacts: knowledge-base
TRUE

CLIPS> (reset)
CLIPS> (run)
        Is the indicator I4 increasing? (yes no) no
        Is the indicator I4 decreasing? (yes no) no
        Is the indicator I4 constant? (yes no) yes
        Is the indicator I4 decreasing? (yes no) no
        All indicators seem to be OK
CLIPS>
```

Figure 2. Example of CLIPS output for manual input.

4. Discussion

The possibility of devising and implementing an AI procedure to extend the fault diagnosis into the realm of prognostic is -at the current state of the art- perfectly possible. It requires a shift from from the "machine thinking" typical of ANN and GA procedures to the "propositional" and "fuzzy" thinking characteristic of real AI (Expert Systems). In the case in point, an application to a real compressor was implemented in [1]. The code performs satisfactorily also in the first stages of condition derangement.

5. Conclusions

The application of expert systems for energy conversion systems failure analysis has been proven to be a reliable and effective aid in dealing with large amount of data and complex fault chains, especially whenever it is essential to understand the consequential process and avoid expensive plant downtimes.

Author Contributions: Conceptualization: E. Sciubba; **Methodology** E. Sciubba, R. Melli; **Software:** E. Sciubba, R. Melli.

Funding: No funding has been received

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Example of PROMISA CLIPS Code for the Compressor Fault Chain

```
;;=====================================================
;;;  Plant Failure Identification and Prognosis Expert System
;;;
;;;    A simple expert system which attempts to identify
;;;    a component failure based on its failure indicator characteristics.
;;;    The knowledge base in this example is a
;;;    collection of facts which represent backward
;;;    chaining rules. CLIPS forward chaining rules are
;;;    then used to simulate a backward chaining inference
;;;    engine.
;;;
;;;    CLIPS Version 6.0
;;;    To execute, merely load, reset, and run.
;;;    Answer questions yes or no.
;;;=====================================================

;;
;;**************************
;;;* INFERENCE ENGINE RULES *
;;;**************************
(defrule propagate-goal ""
  (goal is ?goal)
  (rule (if ?variable $?)
      (then ?goal ? ?value))
  =>
  (assert (goal is ?variable)))

(defrule goal-satified ""
  (declare (salience 30))
  ?f <- (goal is ?goal)
  (variable ?goal ?value)
  (answer ? ?text ?goal)
  =>
  (retract ?f)
  (format t "%s%s%n" ?text ?value))

(defrule remove-rule-no-match ""
  (declare (salience 20))
  (variable ?variable ?value)
  ?f <- (rule (if ?variable ? ~?value $?))
  =>
  (retract ?f))

(defrule modify-rule-match ""
  (declare (salience 20))
  (variable ?variable ?value)
```

```
  ?f <- (rule (if ?variable ? ?value and $?rest))
  =>
  (modify ?f (if ?rest)))



(defrule rule-satisfied ""
  (declare (salience 20))
  (variable ?variable ?value)
  ?f <- (rule (if ?variable ? ?value)
          (then ?goal ? ?goal-value))
  =>
  (retract ?f)
  (assert (variable ?goal ?goal-value)))

(defrule ask-question-no-legalvalues ""
  (declare (salience 10))
  (not (legalanswers $?))
  ?f1 <- (goal is ?variable)
  ?f2 <- (question ?variable ? ?text)
  =>
  (retract ?f1 ?f2)
  (format t "%s " ?text)
  (assert (variable ?variable (read))))

(defrule ask-question-legalvalues ""
  (declare (salience 10))
  (legalanswers ? $?answers)
  ?f1 <- (goal is ?variable)
  ?f2 <- (question ?variable ? ?text)
  =>
  (retract ?f1)
  (format t "%s " ?text)
  (printout t ?answers " ")
  (bind ?reply (read))
  (if (member (lowcase ?reply) ?answers)
   then (assert (variable ?variable ?reply))
        (retract ?f2)
   else (assert (goal is ?variable))))


;;;**************************
;;;* DEFFACTS KNOWLEDGE BASE *
;;;**************************

(deffacts knowledge-base
  (goal is type.failure)
  (legalanswers are yes no)

;;;***************************
;;;* COMPRESSOR FAILURE TYPES *
;;;***************************
;          * STALL *
;
  (rule (if possible_stall is yes)
        (then superfailure is possible_stall))
  (rule (if possible_stall is no)
        (then superfailure is
check_for_possible_chocking))
  (question possible_stall is "Is the indicator I4
increasing?")
;
  (rule (if superfailure is possible_stall and
         confirm.stall is yes)
        (then failure is stall))
  (rule (if failure is stall)
        (then type.failure is compressor_stall))
  (rule (if superfailure is possible_stall and
         confirm.stall is no)
        (then superfailure is
check_for_possible_chocking))
  (question confirm.stall is "Is the indicator I5
decreasing?")
;
;          *CHOCKING *
;
  (rule (if superfailure is
check_for_possible_chocking and
         possible.chocking is yes)
        (then failure is possible_chocking))
  (rule (if superfailure is
check_for_possible_chocking and
         possible.chocking is no)
        (then superfailure1 is
check_for_possible_compressor_fouling))
  (question possible.chocking is "Is the
indicator I4 decreasing?")
;
  (rule (if failure is possible_chocking and
         confirm.chocking is yes)
        (then class is compressor_chocking))
  (rule (if class is compressor_chocking)
        (then type.failure is
compressor_chocking))
  (rule (if failure is possible_chocking and
         confirm.chocking is no)
        (then class is check_I5))
  (question confirm.chocking is "Is the
indicator T3 decreasing?")
;
  (rule (if class is check_I5 and
         confirm.mass_chocking is yes)
        (then order is
compressor_mass_chocking))
  (rule (if order is compressor_mass_chocking)
        (then type.failure is
compressor_mass_chocking))
  (rule (if class is check_I5 and
             confirm.mass_chocking is no)
```

```
            (then superfailure1 is
check_for_possible_compressor_fouling))
  (question confirm.mass_chocking is "Is the
Indicator I5 increasing?")
;
;          *FOULING *
;
 (rule (if superfailure1 is
check_for_possible_compressor_fouling and
possible.compressor_fouling is yes)
       (then failure1 is
possible_compressor_fouling))

 (rule (if superfailure1 is
check_for_possible_compressor_fouling and
possible.compressor_fouling is no)
       (then failure is elsewhere))
  (question possible.compressor_fouling is "Is
the indicator I3    decreasing?")
 (rule (if failure1 is
possible_compressor_fouling and
        confirm.possible_fouling is yes)
       (then class1 is
confirmed_possible_fouling))
 (rule (if failure1 is
possible_compressor_fouling and
        confirm.possible_fouling is no)
       (then class1 is check_I4_if_constant))
  (question confirm.possible_fouling is "Is the
indicator I4 decreasing?")
 (rule (if class1 is confirmed_possible_fouling
and
        confirm.fouling is yes)
       (then order1 is
assess_compressor_fouling))
 (rule (if class1 is confirmed_possible_fouling
and
        confirm.fouling is no)
       (then order1 is elsewhere))
;check_I5))
 (rule (if order1 is assess_compressor_fouling)
       (then type.failure is compressor_fouling))
  (question confirm.fouling is "Is the indicator
I2 constant?")
;
;
  (rule (if class1 is check_I4_if_constant and
           possible.compressor_fouling is yes)
           (then order1 is
confirm.compressor_fouling))
  (rule (if class1 is check_I4_if_constant and
            possible.compressor_fouling is no)
           (then order1 is elsewhere))
    (question possible.compressor_fouling is "Is
the indicator I4 constant?")
```

References

1. PARC is turning 50: from Ethernet and laser printing to this wild new tech; ZDNet April 2020. Online Available: https://www.zdnet.com/article/parc-is-turning-50-from-ethernet-and-laser-printing-to-this-wild-new-tech/ (accessed on 27 February 2020).
2. Biagetti T, Sciubba E. PROMISE: A tool for the on-line intelligent performance prediction of cogeneration plants. In Proceedings of the ECOS02, Berlin, Germany, 3–5 July 2002; pp. 463–471.
3. Biagetti T, Sciubba E. A first step towards unmanned intelligent process management: A procedure for the diag- nostics and prognostics of energy conversion plants. *Int. J. Appl. Thermodyn.* 2002, 5, 85–99.
4. Gülen, S.C.; Griffin, P.R.; Paolucci, S. *Real Time On-Line Performance Diagnostics of Heavy-Duty Industrial Gas Turbines*; ASME paper 2000-GT-312; American Society of Mechanical Engineers; New York, NY, USA, 2000.
5. Roemer, M.J.; Kacprzynski, G.J. *Advanced Diagnostics and Prognostics for Gas Turbine Engine Risk Assessment*; ASME Paper 2000-GT-30; New York, NY, USA, 2000.
6. Tsalavoutas, A.; Aretakis, N.; Mathioudakis, K.; Stamatis, A. *Combining Advanced Data Analysis Methods for the Constitution of an Integrated Gas Turbine Condition Monitoring and Diagnostic System*; ASME Paper 2000-GT-34; New York, NY, USA, 2000.
7. Forsyth, G.; Delaney, J. *Designing Diagnostics Expert Systems for Long-Term Supportability*; ASME Paper, 2000-GT-31; New York, NY, USA, 2000.
8. Ozgur, D.; Lakshminarasimha, A.N.; Rucigay, R.; Morjaria, M.; Sanborn, S. *Remote Monitoring and Diagnostics System for GE Heavy-Duty Gas Turbines*; ASME Paper, 2000-GT-314; New York, NY, USA, 2000.
9. Sciubba, E.; Melli, R. *Artificial Intelligence in Thermal Systems Design: Concepts and Applications*; Nova Science:;New York, NY, USA, 1998.
10. Sriram, R.D. *Intelligent Systems for Engineering*; Springer: New York, NY, USA, 1997.