German Journal of Artificial Intelligence

Automated Computation of Therapies Using Failure Mode and Effects Analysis in the Medical Domain

Malte Luttermann[1] · **Edgar Baake**[2] · **Juljan Bouchagiar**[3] · **Benjamin Gebel**[4] · **Philipp Grüning**[5] · **Dilini Manikwadura**[6] · **Franziska Schollemann**[7] · **Elisa Teifke**[8] · **Philipp Rostalski**[7] · **Ralf Möller**[1]

Received: 21 June 2023 / Accepted: 9 September 2023

Abstract Failure mode and effects analysis (FMEA) is a systematic approach to identify and analyse potential failures and their effects in a system or process. The FMEA approach, however, requires domain experts to manually analyse the FMEA model to derive risk-reducing actions that should be applied. In this paper, we provide a formal framework to allow for automatic planning and acting in FMEA models. More specifically, we cast the FMEA model into a Markov decision process which can then be solved by existing solvers. We show that the FMEA approach can not only be used to support medical experts during the modelling process but also to automatically derive optimal therapies for the treatment of patients.

[1]Institute of Information Systems, University of Lübeck, Germany
E-mail: {luttermann,moeller}@ifis.uni-luebeck.de
[2]Institute of Telematics, University of Lübeck, Germany
E-mail: baake@itm.uni-luebeck.de
[3]Institute for Software Engineering and Programming Languages, University of Lübeck, Germany
E-mail: juljan.bouchagiar@isp.uni-luebeck.de
[4]Department of Infectious Diseases and Microbiology, University Hospital Schleswig-Holstein/ Campus Lübeck, Germany
E-mail: Benjamin.Gebel@uksh.de
[5]Institute for Neuro- and Bioinformatics, University of Lübeck, Germany
E-mail: ph.gruening@uni-luebeck.de
[6]Institute for Molecular Medicine, University of Lübeck, Germany
E-mail: DiliniTharindika.Manikwadura@uksh.de
[7]Institute for Electrical Engineering in Medicine, University of Lübeck, Germany
E-mail: {franziska.schollemann,philipp.rostalski}@uni-luebeck.de
[8]Department of Anesthesiology and Intensive Care, University Hospital Schleswig-Holstein/ Campus Lübeck, Germany
E-mail: Elisa.Teifke@uksh.de

1 Introduction

Failure mode and effects analysis (FMEA) is a widely used framework to assess the risk of a system or process. In particular, FMEA breaks down a system into a hierarchy of components, their functionalities, and potential failures to allow for a systematic analysis of each component or function and their potential failures [24]. Failures are then prioritised according to their potential harm, that is, their severity, their likelihood of occurrence, and their detectability [8]. Based on the priorities of the failures, countermeasures against the most critical failures can be developed [29]. The FMEA approach is an industry standard in the engineering and the manufacturing industry. The application of FMEA helps to improve the design of the final product and to detect and reduce the risk of failure [8,15]. FMEA is already applied during the manufacturing process of medical devices [14] to increase their reliability and helps to satisfy the quality requirements of particular medical processes such as administering drugs to patients [1]. Our goal in this paper is to automate planning and acting in FMEA models. In particular, we apply FMEA to the medical domain to automatically derive optimal therapies for individual patients.

Although there is a lot of work on using formal models for diagnosis and treatment of patients, there are, to the best of our knowledge, no applications of FMEA to model the underlying cause-effect relationships in the human body used in the decision making process of medical experts for diagnosis and treatment. As the human body is without doubt a highly complex system and all of these formal models must be created manually by domain experts, the FMEA approach has the potential to support medical experts during the modelling process of cause-effect relation-

ships in the human body. FMEA ensures compliance with the model hierarchy, thereby yielding a structured model approach that helps to deal with the complexity of the model. Furthermore, as the FMEA approach in its current form requires a lot of manual work from domain experts even after the FMEA model has been constructed because an FMEA model does not provide any automated reasoning capabilities to derive countermeasures (i.e., actions) against potential failures, we propose to cast the FMEA model into a Markov decision process (MDP). An MDP provides a mathematical framework to model sequential decision problems in a fully observable, stochastic environment. In particular, we show how an arbitrary FMEA model can be transformed into an MDP such that existing MDP solvers can be applied to automatically derive optimal policies, thereby allowing us to use the transformation from FMEA model to MDP to fully automate the computation of possible countermeasures for potential failures in the FMEA model. Further, we demonstrate that a policy obtained from the MDP solver can be used to obtain an optimal therapy for an individual patient.

Related work. The FMEA approach is widely applied in various industries such as the automotive industry [22], the aerospace industry [31], and manufacturing industries in general [19]. FMEA supports companies during the design phase of their product and is especially prevalent during the design phase of products with special reliability requirements in safety-critical application environments. In the medical (healthcare) domain, the FMEA approach is commonly applied during the design phase of medical devices [20] as these devices are often used in hospitals and other health-critical environments that must fulfil extraordinary safety requirements. The administration of drugs to patients in a hospital as well as the evaluation of an automated treatment planning tool for radiation are other safety-critical processes in the medical domain where the FMEA approach is already being used [1,13]. However, to the best of our knowledge, the FMEA approach is not yet applied to model the decision-making process of a medical expert for diagnosis and treatment based on the cause-effect relationships in the human body. All of the aforementioned industries rely on manual work of domain experts to not only construct the FMEA model but also to use it to derive actions serving as countermeasures to the potential failures.

Decision support systems (DSSs) are approved by many practitioners in the clinical routine for their support in finding the best action while including a growing amount of information and clinical data ("information overload") [2,5,21,25]. If the standard guidelines are integrated within a DSS, the outcome is improved [18, 21]. Examples of DSS applications, for instance within the mechanical ventilation domain, are the calculation of initial ventilation parameters [12,27], the construction of a weaning protocol for children [9], or the ventilation of patients with acute respiratory distress syndrome [6]. Nevertheless, DSSs often have a very specific and limited use case wherefore each application requires an individual DSS to be elaborated and implemented. In general, there are plenty of works using various mathematical frameworks to establish medical decision support [28]—for example, partially observable MDPs (a generalisation of MDPs) are already employed to support in diagnosis and treatment [3,10,32].

Our contributions. We first extend the standard definition of an FMEA model by adding variables (parameters) to functions. The variables and their qualitative relationships among each other allow us to define a formal semantics of failures and actions in an FMEA model, i.e., failures indicate that the value of a variable is outside of its normal range and an action restricts the set of possible values for the variables. Having defined a formal semantics for an FMEA model, we next show how such a model can be transformed into an MDP such that all transition probabilities and rewards can be directly derived from the FMEA model. To obtain the possible successor states in the MDP, we apply qualitative causal reasoning in the FMEA model. The MDP can then be solved using existing MDP solvers to obtain an optimal policy, which maps each possible state of the system to the best possible action for that particular state. We present an algorithm to automatically derive the best possible therapy according to the initial FMEA model for a particular patient using the optimal policy obtained by solving the MDP.

Structure of this paper. The remaining part of this paper is structured as follows. Section 2 introduces the necessary background information for the main part of this paper. We first define FMEA models and then introduce MDPs as a mathematical framework for modelling sequential decision problems in a fully observable, stochastic environment. Afterwards, in Section 3, we show how an FMEA model can be transformed into an MDP which can then be used for automated planning and acting in an FMEA model. We introduce qualitative causal reasoning to obtain an algorithm that computes the possible successor states after applying an action in the MDP. Section 4 introduces an algorithm to compute the best possible therapy for a given patient according to a given FMEA model. Finally, we discuss applications and limitations of our approach in Section 5 before we conclude this paper in Section 6.

2 Preliminaries

In this section, we introduce the necessary background information for the remainder of this paper. We begin by formally defining the syntax of an FMEA model, which we then extend by adding variables to the FMEA model to obtain an extended FMEA model for which we can define a formal semantics. Afterwards, we define MDPs as a framework to model sequential decision problems in a fully observable, stochastic environment.

2.1 Failure Mode and Effects Analysis

FMEA is a systematic approach to identify and analyse potential failures in a system or process [24]. During the FMEA process, the system is decomposed into its components and functions and for each function, the possible failures are identified. Every failure is assigned its potential severity, its likelihood of occurrence, and its detectability, which are then combined into a *risk priority number* to assess the risk of the failure. To be able to apply the FMEA approach to the medical domain to automatically derive therapies for particular patients, we begin by defining an FMEA model.

Definition 1 (FMEA Model) An *FMEA model* is defined as a tuple $\mathcal{F} = (C, F, E, A, C2C, F2F, E2E, C2F, F2E, A2E, RP, AP)$, where

- C is a finite set of components,
- F is a finite set of functions,
- E is a finite set of failures,
- A is a finite set of actions,
- $C2C \subseteq C \times C$ is the component hierarchy,
- $F2F \subseteq F \times F$ is the function hierarchy,
- $E2E \subseteq E \times E$ is the failure hierarchy,
- $C2F \subseteq C \times F$ assigns functions to components,
- $F2E \subseteq F \times E$ assigns failures to functions,
- $A2E \subseteq A \times E2E$ assigns actions to failure pairs,
- $RP \subseteq E \times \{1, \dots, 10\} \times \{1, \dots, 10\} \times \{1, \dots, 10\}$ assigns each failure a severity, occurrence, and detectability value, and
- $AP \subseteq A \times \{d, p\}$ specifies the type of each action ("d" for detective or "p" for preventive).

Note that in its current form, an FMEA model is only used by domain experts when thinking about potential risks of a system or process, i.e., the model itself does not "do" anything except for being visually displayed in some kind of graphical user interface. The hierarchy of components, functions, and failures, induced by $C2C$, $F2F$, and $E2E$, respectively, is constrained to form a connected directed tree (polytree), i.e., a directed acyclic graph (DAG) where any two vertices are connected by exactly one path when replacing the directed edges by undirected edges. Additionally, each component is restricted to be a sub-component of at most one other component but it is allowed for a component to have multiple sub-components (analogously for functions and failures)—that is, $C2C$, $F2F$, and $E2E$ are left-functional ($1{:}N$) relations. Further, the relations $C2F$, $F2E$, and $A2E$ are right-total ($N{:}1$), meaning that each function is attached to exactly one component in $C2F$, each failure is attached to exactly one function in $F2E$, and each action is attached to exactly one pair of failure cause and failure effect in $A2E$. Note that the other way around, there is no restriction, e.g., every component can have arbitrarily many (including zero) functions attached to it. In general, the idea is that each component supplies one or more functionalities (functions) and each function might go wrong in one or more ways (failures). Actions are used as a remedy to deal with failures (i.e., functions going wrong). The first entry in each tuple in RP serves as a key such that there is exactly one tuple contained in RP for each failure $e \in E$, assigning risk parameters (severity, occurrence, and detectability) to e. By $\text{sev}(e)$, $\text{occ}(e)$, and $\text{det}(e)$ we denote the severity, occurrence, and detectability of e, respectively. We refer to the causes of a failure e by $\text{causes}(e) = \{e' \in E \mid (e', e) \in E2E\}$, and denote its effects by $\text{effects}(e) = \{e' \in E \mid (e, e') \in E2E\}$. Analogously to RP, there is exactly one tuple contained in AP for each action $a \in A$, assigning a type ("d" for detective or "p" for preventive) to a.

Example 1 (FMEA Model) Consider the FMEA model illustrated in Fig. 1. There are two components "Perialveolar interstitium" (denoted as c_1) and "Respiratory system" (c_2), and the arrow $c_1 \to c_2$ indicates that c_1 is a sub-component of c_2 (analogously, f_1 is a sub-function of f_2 and e_1 is a cause for e_2). More specifically, the set of components is given by $C = \{c_1, c_2\}$, the set of functions is given by $F = \{f_1, f_2\}$, the set of failures is given by $E = \{e_1, e_2\}$, and the set of actions is given by $A = \{d_1, p_1\}$. The hierarchies for components, functions, and failures are given by $C2C = \{(c_1, c_2)\}$, $F2F = \{(f_1, f_2)\}$, and $E2E = \{(e_1, e_2)\}$, respectively. The remaining relations are given by $C2F = \{(c_1, f_1), (c_2, f_2)\}$, $F2E = \{(f_1, e_1), (f_2, e_2)\}$, and $A2E = \{(d_1, (e_1, e_2)), (p_1, (e_1, e_2))\}$. Finally, the risk parameters are given by $RP = \{(e_1, 5, 4, 9), (e_2, 7, 5, 9)\}$ and the action parameters are given by $AP = \{(d_1, d), (p_1, p)\}$.

As the FMEA model in its current form is not able to do anything, we next extend an FMEA model by adding variables (parameters) to functions and afterwards define the semantics of the extended FMEA model.

Definition 2 (Extended FMEA Model) An *extended FMEA model* is defined as a tuple $\mathcal{F} = (C, F, E,$

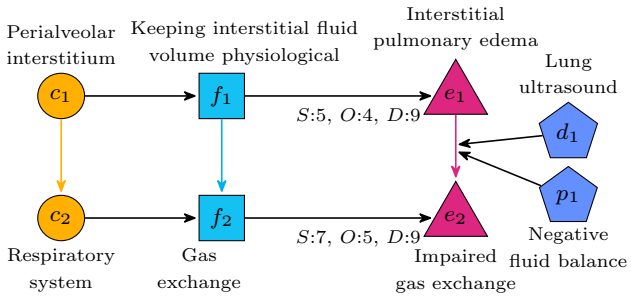

Fig. 1: An example for an FMEA model. The components C are given by the circles, the functions F by the rectangles, the failures E by the triangles, and the actions A by the pentagons. The hierarchy relations and assignment relations are indicated by the edges between the components, functions, failures, and actions, respectively. The name of the actions indicate the action parameters AP ("d" for detective and "p" for preventive) and the risk parameters RP are given by the S, O, and D values next to the failures.

$A, C2C, F2F, E2E, C2F, F2E, A2E, RP, AP, pre, post, V, F2V, \mathcal{G}$), where C, F, E, A, $C2C$, $F2F$, $E2E$, $C2F$, $F2E$, $A2E$, RP, and AP form an FMEA model according to Definition 1,

- pre assigns preconditions in form of Boolean expressions to actions,
- $post$ assigns postconditions in form of Boolean expressions to actions,
- V is a finite set of variables,
- $F2V \subseteq F \times V$ assigns variables to functions, and
- $\mathcal{G} = (V, E')$ is a directed graph that encodes qualitative relationships between the variables in V.

An action $a \in A$ is called applicable in state s if s satisfies the preconditions of a. Sometimes, an action a might invoke side effects which are captured by the postconditions of a. The relation $F2V$ is left-total (1:N), meaning that each variable is attached to exactly one function. We require each function to have at least one variable attached to it but there is no limit of variables being attached to the same function. The idea behind adding variables to functions is that each function $f(v_1, \ldots, v_m) = (v'_1, \ldots, v'_k)$ produces an output value for each variable $v'_1, \ldots, v'_k$ ($k \geq 1$) attached to f. The purpose of variables is that functions are characterised by the variables attached to them, allowing us to define a formal semantics for failures and actions. A function f is not required to take any variable as an input (i.e., $m = 0$ is allowed). If there are input variables $v_1, \ldots, v_m$ for a function f, $v_1, \ldots, v_m$ are outputs of sub-functions of f. Moreover, the associated graph $\mathcal{G} = (V, E')$ models the qualitative relationships between all variables in V. In particular, $E' \subseteq V \times V \times \{+, -, ?\}$ is a set of labelled edges, that is, there is an edge $u \xrightarrow{\ell} v$ in $\mathcal{G}$ if $(u, v, \ell) \in E'$. An edge $u \xrightarrow{+} v$ encodes that increasing u will yield an increase of v, $u \xrightarrow{-} v$ means that increasing u will yield a decrease of v, and $u \xrightarrow{?} v$ entails that the effect of u on v is unknown. If a label ℓ is irrelevant in a specific context, we omit it and simply write $u \to v$ instead of $u \xrightarrow{\ell} v$. Vertices that are connected by an edge are called adjacent and are neighbours of each other. $\mathrm{Pa}(v)$ denotes the set of parents of a variable v, i.e., $\mathrm{Pa}(v) = \{u \mid \exists \ell : (u, v, \ell) \in E'\}$ and the children of v are given by $\mathrm{Ch}(v) = \{u \mid \exists \ell : (v, u, \ell) \in E'\}$. By $\mathrm{range}(v)$ we denote the set of possible values a variable $v \in V$ can take. For simplicity, we assume that $\mathrm{range}(v) \subseteq \{\mathrm{tooLow}, \mathrm{normal}, \mathrm{tooHigh}\}$ and $\mathrm{normal} \in \mathrm{range}(v)$ for all variables v throughout this paper, i.e., the value of each variable can either be in its normal range or deviate from its normal range in both directions. Consequently, each failure $e \in E$ has either the form $e := \mathrm{left\_critical}(v_i)$ (i.e., implying that $v_i = \mathrm{tooLow}$) or $e := \mathrm{right\_critical}(v_i)$ (i.e., implying that $v_i = \mathrm{tooHigh}$), where $v_i \in V$ is a variable attached to the function to which e is attached. For example, if there is a variable v_i called "body temperature" and a failure $e := \mathrm{right\_critical}(v_i)$ (fever), then v_i can either be assigned the value normal or tooHigh, while $v_i = \mathrm{tooHigh}$ triggers the failure e. However, different ranges (and hence different failure semantics) are also possible and do not affect our approach to automate planning and acting in an FMEA model.

Example 2 (Extended FMEA Model) Take a look at the extended FMEA model depicted in Fig. 2 which builds on the FMEA model from Fig. 1. There are now variables $V = \{v_1, v_2\}$ attached to the functions f_1 and f_2, respectively. In particular, the assignments of variables to functions are given by $F2V = \{(f_1, v_1), (f_2, v_2)\}$. The qualitative relationships between the variables in V are encoded by the graph $\mathcal{G} = (V, \{(v_1, v_2, -)\})$. To apply p_1, there is a precondition that an interstitial pulmonary edema must be detected first, i.e., $pre = \{(p_1, v_1 = \mathrm{tooHigh})\}$. There are no side effects (postconditions) for both of the actions, that is, $post = \emptyset$. The model states that too much interstitial fluid volume results in an interstitial pulmonary edema, i.e., $e_1 = \mathrm{right\_critical}(v_1)$, and too little diffusing capacity of the lung impairs the gas exchange, i.e., $e_2 = \mathrm{left\_critical}(v_2)$. The edge $v_1 \xrightarrow{-} v_2$ implies that if the interstitial fluid volume is too high, the diffusing capacity of the lung will eventually become too low.

From now on, we focus on extended FMEA models and simply write FMEA model instead of extended FMEA

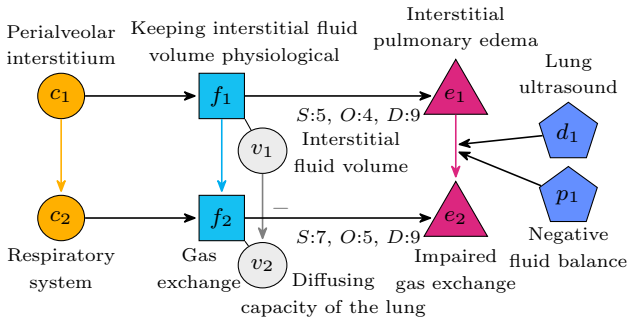

Fig. 2: An example for an extended FMEA model building on the FMEA model illustrated in Fig. 1. Now, there is a variable attached to each function in the model and the qualitative relationship between the two variables is encoded by the labelled edge between them.

model to refer to an FMEA model as defined in Definition 2. Next, we define the semantics of an FMEA model whose goal is to assess the risk of a system. It is important to differentiate between the risk at class level (e.g., the risk of the human body itself) and the risk at instance level (e.g., the risk of a specific human individual). At class level, we are interested in reducing the risk of the system by changing the model itself, e.g., by adding actions to it. For example, if a system contains a severe failure that cannot be detected at all, the total risk of the system decreases as soon as a detection action is added to the model to make that failure detectable. The semantics of an FMEA model $\mathcal{F}$ at class level can therefore be defined as

$$\text{risk}(\mathcal{F}) = \max_{e \in E} \phi(\text{sev}(e), \text{occ}(e), \text{det}(e)),$$

where $\phi : \{1, \ldots, 10\} \times \{1, \ldots, 10\} \times \{1, \ldots, 10\} \to \{\text{green}, \text{orange}, \text{red}\}$ is a total function (i.e., ϕ is defined for all possible combinations of severity, occurrence, and detectability values) mapping the severity, occurrence, and detectability of a failure to a risk value for that failure. We require $\{\text{green}, \text{orange}, \text{red}\}$ to be an ordered set (given in ascending order), i.e., $\max\{\text{green}, \text{orange}\} = \text{orange}$ and $\max\{\text{orange}, \text{red}\} = \text{red}$. The risk of an FMEA model at class level is therefore the risk of the most critical failure in the model. Other definitions for the semantics of an FMEA model are possible as well.

For the remaining part of this paper, we focus on assessing the risk of an FMEA model at instance level. At instance level, we are interested in determining a sequence of actions that are actually executed to reduce the risk of a particular instance. For example, in the medical domain, we aim to compute a therapy (sequence of actions) for a particular patient (instance). Instantiating an FMEA model for a particular instance yields a state s determined by the possible values each variable $v \in V$ can take. For example, if it is known that a patient has fever, the variable "body temperature" is assigned the value "tooHigh". Applying an action yields a new state and each state is assigned a risk value based on failures that can possibly occur in that state. The goal is to minimise the risk by performing a sequence of actions to reach a state having a low risk value (i.e., a state corresponding to the patient being healthy). Before we formally define states and the risk of a state in Section 3, we lay the foundations to automate planning and acting in an FMEA model—that is, to automatically compute the best possible therapy for a specific patient according to the FMEA model.

2.2 Markov Decision Processes

An MDP [4] is a mathematical framework for modelling a sequential decision problem with discrete time and a fully observable, stochastic environment with a Markovian transition model and additive rewards (i.e., there is a reward in each state and these rewards are added up for the sequence of states that have been visited).

Definition 3 (MDP) We define an *MDP* as a tuple $\mathcal{M} = (S, A, s_0, P, R, \gamma)$, where

- S is a finite set of states,
- A is a finite set of actions,
- $s_0 \in S$ is the initial state,
- $P : S \times A \times S \to [0, 1]$ is the transition function, i.e., $P(s, a, s')$ yields the probability of transitioning into state s' when taking action a in state s,
- $R : S \times A \times S \to \mathbb{R}$ is a reward function, i.e., $R(s, a, s')$ is the reward for transitioning to state s' when taking action a in state s, and
- $\gamma \in [0, 1]$ is the discount factor.

The discount factor γ indicates how much future rewards should be discounted, e.g., $\gamma = 1$ weights all rewards equally while smaller values for γ render future rewards less significant. We write $P(s' \mid s, a)$ to refer to the probability of transitioning to state s' when taking action a in state s ($\sum_{s' \in S} P(s' \mid s, a) = 1$).

Example 3 (MDP) Figure 3 shows an exemplary MDP with states $S = \{s_1, s_2\}$ and actions $A = \{a_1, a_2\}$, depicted as a state-transition system. The initial state is s_1. In each state, both actions can be applied and the transition probabilities are written next to the edges. For example, when applying action a_1 in state s_1, the probability to remain in state s_1 is 0.3 and the probability to transition to state s_2 is 0.7. In this particular example, $2 \cdot 2 \cdot 2 = 8$ rewards need to be specified to define the reward function ($R(s_1, a_1, s_1)$, $R(s_1, a_1, s_2)$,

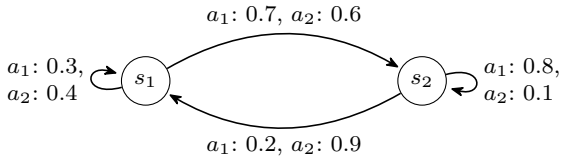

Fig. 3: An example for an MDP with states $S = \{s_1, s_2\}$ and actions $A = \{a_1, a_2\}$. The transition probabilities are given by the numbers written next to the edges. Rewards and the discount factor are omitted for brevity.

and so on). We omit the discount factor and the exact specification of the reward function for brevity.

A *policy* $\pi : S \to A$ is a total function which maps each state to an action, i.e., $\pi(s)$ returns the action to take in state s. Solving the MDP yields an optimal policy π^* which maps every state to the best possible action to take in that state. The optimal sequence of actions is the one having the maximum expected reward, i.e., the optimal policy depends on the choice of the transition probabilities and reward function.

Example 4 (Policy) Consider again the MDP depicted in Fig. 3. A possible policy is

$$\pi(s) = \begin{cases} a_1 & \text{if } s = s_1 \\ a_2 & \text{if } s = s_2. \end{cases}$$

The optimality of π depends on the choice of R.

In the next section, we show how an arbitrary FMEA model can be transformed into an MDP, which can then be solved to obtain an optimal policy, allowing us to automatically compute the best possible action to take in a specific state of the model. We also introduce qualitative causal reasoning to compute the successor states after applying an action in a particular state.

3 Automated Planning and Acting in FMEA Models Using MDPs

In this section, we show how to automatically compute the best possible sequence of actions for a specific instance of an FMEA model. To automatically obtain the best possible sequence of actions, we construct an MDP from an FMEA model. The MDP can then be solved, yielding an optimal policy for decision making.

3.1 Construction of the Markov Decision Process

Given an FMEA model $\mathcal{F} = (C, F, E, A, C2C, F2F, E2E, C2F, F2E, A2E, RP, AP, pre, post, V, F2V, \mathcal{G})$, we construct an MDP $\mathcal{M} = (S, A, s_0, P, R, \gamma)$ as follows.

State space. The state space S is defined by the possible values each variable in $V = \{v_1, \ldots, v_n\}$ can take. More specifically, $S \subseteq \times_{i=1}^{n} 2^{\text{range}(v_i)}$ with 2^X denoting the power set of X without the empty set. As there is typically no evidence available in the beginning, the initial state s_0 defaults to $s_0 = \langle \text{range}(v_1), \ldots, \text{range}(v_n) \rangle$. However, it is also conceivable to start with a different initial state if there is additional evidence available. Note that we model the states in a way such that every state is observable, even though there is still uncertainty about the exact values of the variables. We denote by $\text{poss}_s(v_i)$ the set of possible values for variable v_i in state s (that is, $\text{poss}_s(v_i)$ references the i-th component of the state vector s).

Example 5 Consider the set of variables $V = \{v_1\}$ with $\text{range}(v_1) = \{\text{tooLow}, \text{normal}, \text{tooHigh}\}$. The state space of the MDP then consists of seven different states and is given by $S = \{s_1, \ldots, s_7\}$ where $s_1 = \langle\{\text{tooLow}\}\rangle$, $s_2 = \langle\{\text{normal}\}\rangle$, $s_3 = \langle\{\text{tooHigh}\}\rangle$, $s_4 = \langle\{\text{tooLow}, \text{normal}\}\rangle$, $s_5 = \langle\{\text{tooLow}, \text{tooHigh}\}\rangle$, $s_6 = \langle\{\text{normal}, \text{tooHigh}\}\rangle$, and $s_7 = \langle\{\text{tooLow}, \text{normal}, \text{tooHigh}\}\rangle$. Each state indicates the possible ranges of the variables in V, e.g., s_1 is the state where it is known that $v_1 = \text{tooLow}$, s_4 is the state where it is known that $v_1 \neq \text{tooHigh}$ but there is no information about whether $v_1 = \text{tooLow}$ or $v_1 = \text{normal}$, and so on. Without any evidence, the initial state is s_7, where v_1 may take any value.

Action space and transition probabilities. The set of actions A in the MDP is directly given by the set of actions from the FMEA model. Both the preconditions and the postconditions for the actions are directly transferred to the MDP as well, with an additional precondition $|\text{poss}_s(v_i)| > 1$ being added to each detection action a that is used to detect the value of variable v_i in state s, i.e., a is only applicable if the value that should be detected is not already known. If an action $a \in A$ cannot be applied in state s (i.e., its preconditions are not met), we set $P(s' \mid s, a) = 0$ for all successor states s'. Otherwise, let $S' \subseteq S$ be the set of possible successor states after applying action a in state s and let $k = |S'|$. Note that for a particular instance, there is exactly one successor state when applying action a in state s but in the general MDP, all possible successor states that are reachable for any instance need to be considered. The computation of possible successor states relies on the qualitative relationships encoded in the graph $\mathcal{G}$ and is described in detail in Section 3.2. Given the set of possible successor states and the probability p for an action a to be applied successfully, we set $P(s' \mid s, a) = p/k$ for all successor states $s' \in S'$ with $s' \neq s$ (uniform distribution—can also be adjusted if additional information is available). The probability p of a detection

action a (i.e., $(a,d) \in AP$) is given by $\text{prob}(D) = (9-(D-1))/9$ where $D \in \{1,\ldots,10\}$ is the detectability of the failure e' such that $(a,(e',e)) \in A2E$ (i.e., a is attached to the failure pair $(e',e) \in E2E$). Note that other probability measures are possible as well (e.g., $\text{prob}(D) = (10-D)/10$). For a prevention action a (i.e., $(a,p) \in AP$), we set $p = \text{prob}(O)$ where $O \in \{1,\ldots,10\}$ is the occurrence of the failure e' such that $(a,(e',e)) \in A2E$. Moreover, if the application of an action a in state s is not successful, the system remains in state s, i.e., $P(s \mid s,a) = 1-p$, given that a has no postconditions attached to it. In case an action has postconditions attached to it, they are incorporated into the state regardless of whether the action has been successful, i.e., if an action fails, only the effect of the action itself is not incorporated into the new state whereas the postconditions are. Finally, for all states $s'' \notin S'$ that are not reachable when applying action a in state s, we set $P(s'' \mid s,a) = 0$.

Example 6 Consider an FMEA model with two failures $e_1 := \text{right\_critical}(v_1)$ and $e_2 := \text{right\_critical}(v_2)$ (i.e., $\text{range}(v_1) = \text{range}(v_2) = \{\text{normal}, \text{tooHigh}\}$) with $(e_1,e_2) \in E2E$ and a prevention action a attached to (e_1,e_2) (without preconditions). Let $s = \langle\{\text{tooHigh}\}, \{\text{normal}\}\rangle$. Then, the set of possible successor states after applying action a to prevent e_1 is given by $S' = \{s'\}$ with $s' = \langle\{\text{normal}\},\{\text{normal}\}\rangle$. Assuming that a always succeeds, applying a sets $\text{occ}(e_1) = 1$ as a prevents e_1 from occurring (more details on the exact definition of action semantics are given in Section 3.2). In consequence, we obtain $P(s' \mid s,a) = \text{prob}(1)/1 = (9-(1-1))/9 = 1$ as well as $P(s \mid s,a) = 0$, i.e., the action sets $v_1 = \text{normal}$ while v_2 is left unchanged.

Reward function. The reward for entering the initial state s_0 is set to zero, i.e., $R(s,a,s_0) = 0$ for all actions a and states s. For all successor states $s' \neq s_0$, we define the reward $R(s,a,s')$ for going from state s to successor state s' with action a as described below. Each state s' induces a set of failures that cannot be ruled out in s', e.g., if $\text{tooLow} \in \text{poss}_{s'}(v_i)$, a failure $e := \text{left\_critical}(v_i)$ corresponding to v_i being too low cannot be ruled out (analogously for tooHigh and failures being right_critical). Let $E_{s'}$ denote the set of failures that cannot be ruled out in state s' and let $0 \leq RPN_e \leq 1000$ denote the risk priority number for the failure e. Then, we define the reward for going from any state s to successor state s' with action a as

$$R(s,a,s') = \frac{1}{|E_{s'}|} \sum_{e \in E_{s'}} p_e \cdot (1000 - RPN_e),$$

where p_e is the failure probability for the failure e (if failure probabilities are unknown, p_e can be set to one for every failure e). If $|E_{s'}| = 0$, we set $R(s,a,s') = \infty$. Note that neither s nor a occur in the right-hand side of the equation, i.e., the reward for changing into state s' does not depend on the previous state s and the performed action a. However, we include both s and a into the left-hand side of the equation to demonstrate that a more fine-grained definition of the reward function is also conceivable if the necessary information is available. The maximum value for the risk priority number RPN_e is 1000 and RPN_e is defined as

$$RPN_e = \begin{cases} \min\limits_{e' \in \text{causes}(e)} RPN_{e'} & \text{if causes}(e) \neq \emptyset \\ 0 & \text{otherwise,} \end{cases}$$

where the risk priority number for each cause e' of e is a product of severity, occurrence, and detectability values. More specifically, for a cause-effect pair (e',e), we have $RPN_{e'} = \text{sev}(e) \cdot O' \cdot D'$ with $D' = \det(e')$ if there exists a detection action for (e',e) which is applicable in s' and otherwise $D' = 10$, and $O' = \text{occ}(e')$ if there exists a prevention action attached to (e',e) whose effect is already manifested in s' and otherwise $O' = 10$ (recall that 10 is the maximum possible number both for the detectability and the occurrence). The idea is that the risk of each state s' depends on the detectability and the treatability of the failures that cannot be ruled out in s', i.e., if a failure can neither be detected nor treated, it has a high risk priority number assigned to it. The minimum operator corresponds to a conjunction (AND) of failure causes. Clearly, other operators such as a disjunction (OR) of failure causes could be used as well (i.e., max instead of min for the computation of RPN_e). The choice of the discount factor γ is not part of the transformation from FMEA model to MDP as γ is set by the user independent of the FMEA model.

Example 7 Consider again the FMEA model consisting of two failures $e_1 := \text{right\_critical}(v_1)$ and $e_2 := \text{right\_critical}(v_2)$ with $(e_1,e_2) \in E2E$. It holds that $\text{range}(v_1) = \text{range}(v_2) = \{\text{normal}, \text{tooHigh}\}$ and there is a detection action a attached to (e_1,e_2) (with the default precondition $|\text{poss}_s(v_1)| > 1$ in each state s). Further, let $\text{sev}(e_1) = 6$, $\text{occ}(e_1) = 5$, $\det(e_1) = 9$, $\text{sev}(e_2) = 8$, $\text{occ}(e_2) = 4$, and $\det(e_2) = 9$ and $p_{e_1} = p_{e_2} = 1$ for simplification. Then, we have, for example, $R(s,a,\langle\{\text{tooHigh}\},\{\text{tooHigh}\}\rangle) = \frac{1}{2} \cdot ((1000-0) + (1000 - 8 \cdot 10 \cdot 10))$ (note that $D' = 10$ as a is not applicable due to its precondition and $O' = 10$ because there exists no prevention action in this example) and $R(s,a,\langle\{\text{normal}\},\{\text{normal}\}\rangle) = \infty$ for all states s.

Before we continue to present an algorithm to automatically compute the best therapy for a patient using the optimal policy of the MDP, we first describe how

the possible successor states after the application of an action are computed using qualitative causal reasoning.

3.2 Computation of Successor States

As failures influence other failures, an action might have an effect not only on the failure e it acts on, but also on other failures that are effected by e. Therefore, we propagate the effect of an action through the failure hierarchy to determine the possible successor states after applying an action a in state s. Before we describe how the effect of an action is propagated through the failure hierarchy using *qualitative causal reasoning*, we first define the semantics of an action.

We begin by defining the semantics of a detection action a in state s, assuming that a is applicable in s. Action a is attached to a failure pair $(e', e) \in E2E$ and detects whether the failure cause e' is present. Let $e' = \text{left\_critical}(v_i)$ ($\text{right\_critical}(v_i)$, respectively). Then, it holds that $\text{poss}_{s'}(v_i) = \{\text{tooLow}\}$ ($\{\text{tooHigh}\}$) or $\text{poss}_{s'}(v_i) = \{\text{normal}\}$ after transitioning into state s' by applying action a in state s—that is, a detection action determines the value of a variable v_i if it succeeds (a always succeeds if $\det(e') = 1$). In case $\det(e') > 1$, a might fail occasionally as we have seen earlier at the construction of the transition probabilities. Consequently, after applying a detection action a in state s, it holds that $\text{poss}_{s'}(v_i) \subseteq \text{poss}_s(v_i)$ for all variables v_i and successor states s' because the detection action reduces the uncertainty about the values of the variables. Note that in the real world, it might be possible for a detection action a to return an incorrect value (false positive or false negative) if there is a measurement error. In its current form, the MDP does not account for such measurement errors, i.e., the measured value defines the successor state without taking into account that the measured value might be erroneous.

A prevention action a is attached to a failure pair $(e', e) \in E2E$ as well and prevents the failure cause e' from occurring. More specifically, if $e' = \text{left\_critical}(v_i)$ ($\text{right\_critical}(v_i)$, respectively), then applying action a in state s ensures that $\text{poss}_{s'}(v_i) = \{\text{normal}\}$ after transitioning into state s'. In other words, a prevention action a eliminates the failure cause e' by assigning the value of the corresponding variable v_i to its normal range and hence, we set $\text{occ}(e') = 1$ after applying action a (that is, we assume that the application of a is always successful in preventing e'—it is also conceivable that a might fail sometimes which can be modelled by setting $\text{occ}(e')$ to a value greater than one).

Whenever an action is applied successfully, it might affect not only the failure it directly operates on but also other failures in the failure hierarchy. For example, if we have $(e_1, e_2) \in E2E$ (i.e., e_1 causes e_2) and an action is applied that prevents e_1 from occurring, then e_2 cannot occur as well if there are no other causes for e_2 other than e_1. Analogously, if a detection action determines that e_1 is not present in a particular state s, then e_2 cannot occur in s as well if there are no other causes for e_2 other than e_1. As the presence or absence of a failure might influence the information available about its effects, we employ qualitative reasoning [7] to obtain the possible successor states after applying action a in state s. However, we cannot just apply qualitative reasoning as it is proposed in the literature [7,30] because the propagation does not take into account the causal structure of the failure hierarchy.

Algorithm 1: Compute Successor States

```
1  function succ_states(G = (V, E'), a, s)
2      S' ← ∅;
       // v_i is the variable a acts on
3      foreach possible outcome v_i = r_i of a do
4          s' ← s;
5          if r_i = tooLow then
6              σ ← '−';
7          else if r_i = tooHigh then
8              σ ← '+';
9          else
10             σ ← '0';
11         E'' ← E' \ {(u, v_i, ℓ) | ∃ℓ : (u, v_i, ℓ) ∈ E'};
12         signs ← propagate(G' = (V, E''), s, v_i, σ);
13         foreach (v, σ') ∈ signs do
14             if σ' = '−' ∧ tooLow ∈ range(v) then
15                 s'[v] ← {tooLow};
16             else if σ' = '+' ∧ tooHigh ∈ range(v)
                   then
17                 s'[v] ← {tooHigh};
18             else if σ' = '0' then
19                 s'[v] ← {normal};
20         push(s', S');
21     return S';
```

Instead, we introduce qualitative *causal* reasoning to propagate changes only along the causal directions of the edges in the failure hierarchy. In particular, when intervening on a specific failure e, all incoming edges of e must be cut off before the changes are propagated through the graph [16,17]. Algorithm 1 depicts the algorithm that is used to compute the set of possible successor states S' when applying an action a in state s. The algorithm considers every possible outcome of the action a in state s. For a prevention action, there is a single outcome per definition, i.e., the value of the corresponding variable is set to a specific value in its range. The outcome of a detection action, however, is not known when solving the MDP as multiple outcomes are possible in practice (e.g., detecting the value of a variable might either yield "tooLow" or "normal" and

Algorithm 2: Qualitative Reasoning (based on the qualitative sign propagation algorithm proposed by Druzdzel and Henrion [7])

```
1  function propagate(G = (V, E'), s, u, σ)
2      signs ← empty dictionary;
3      vis ← ∅;
4      foreach v ∈ V do
5          signs[v] ← sign_s(v);
6      propagate_rec(G, u, u, σ, signs, vis);
7      return signs;
8  function propagate_rec(G, u, v, σ, signs, vis)
9      msgs = {σ};
10     signs[v] ← '0';
11     foreach w ∈ Pa(v) \ {u} do
12         ℓ ← label of the edge between w and v;
13         msgs ← msgs ∪ {signs[w] ⊗ ℓ};
14     σ' ← signs[v];
15     foreach m ∈ msgs do
16         σ' ← σ' ⊕ m;
17     signs[v] ← σ';
18     vis ← vis ∪ {v};
19     foreach w ∈ Ch(v) do
20         ℓ ← label of the edge between v and w;
21         m ← signs[v] ⊗ ℓ;
22         if w ∉ vis ∧ signs[w] ≠ m then
23             propagate_rec(G, v, w, m, signs, vis);
```

we do not know in advance which of these will be detected for a particular instance). We denote by r_i the outcome of action a on variable v_i, i.e., a acts on v_i and we have r_i = normal if a is a prevention action, otherwise r_i equals the detected value ($r_i \in \{\text{tooLow}, \text{normal}, \text{tooHigh}\}$). Thus, it holds that $v_i = r_i$ after applying action a, i.e., the i-th component of the successor state vector is set to $\{r_i\}$. We abuse notation and write $s'[v_i]$ instead of $s'[i]$ in the following to refer to the position of a variable v_i in the state vector. The value r_i of v_i is converted to a sign ('−' for "tooLow", '+' for "tooHigh", and '0' for "normal") which is then propagated through a modified version of the graph $\mathcal{G}$ (which encodes the qualitative relationships between the variables using edges labelled with signs '+', '−', and '?') where the edges between the parents of v_i and v_i are removed. The removal of edges between the parents of v_i and v_i in $\mathcal{G}$ yields a modified graph $\mathcal{G}'$ which is then used as an input for the qualitative reasoning algorithm illustrated in Algorithm 2 to avoid the propagation of changes against the causal edge direction.

The algorithm for qualitative reasoning builds on the qualitative sign propagation algorithm proposed by Druzdzel and Henrion [7]. It starts by assigning each variable $v \in V$ a sign in Line 5, depending on the possible values v can take in state s. In particular, we define

$$\text{sign}_s(v) = \begin{cases} \text{'+'} & \text{if } \text{poss}_s(v) = \{\text{tooHigh}\} \\ \text{'−'} & \text{if } \text{poss}_s(v) = \{\text{tooLow}\} \\ \text{'0'} & \text{if } \text{poss}_s(v) = \{\text{normal}\} \\ \text{'?'} & \text{otherwise.} \end{cases}$$

The sign of the variable v_i on which the action a has been applied is updated first by calling *propagate_rec* with $u = v = v_i$ as parameter in Line 6. The propagated information (i.e., the signs of the variables) is stored in a dictionary *signs* and already visited variables are stored in a set *vis* such that each variable is visited at most once. During the propagation procedure, we make use of the sign multiplication ($\otimes$) and sign addition ($\oplus$) operators [30], which are defined in Table 1. In every call of *propagate_rec*, u propagates its sign to v. During the propagation from u to v, all other parents of v are also considered (Line 9 to Line 17). More specifically, the algorithm computes a message from all parents of v to v (note that the message from u is already given by σ) and afterwards uses the sign addition operator to combine the messages of the parents into a new sign for v. After the sign of v has been updated, v propagates its new sign to its children that have not been visited yet by recursively calling *propagate_rec* (Line 19 to Line 23).

Example 8 Consider the graph shown in Fig. 4 with edges $v_1 \xrightarrow{+} v_3$ and $v_2 \xrightarrow{-} v_3$ and assume that v_1 is assigned the sign '+' which is then propagated to its children, i.e., to v_3. If we also have evidence for v_2 suggest-

⊗	+	−	0	?
+	+	−	0	?
−	−	+	0	?
0	0	0	0	0
?	?	?	0	?

⊕	+	−	0	?
+	+	?	+	?
−	?	−	−	?
0	+	−	0	?
?	?	?	?	?

Table 1: Definition of the sign multiplication ($\otimes$) and sign addition ($\oplus$) operators according to Wellman [30].

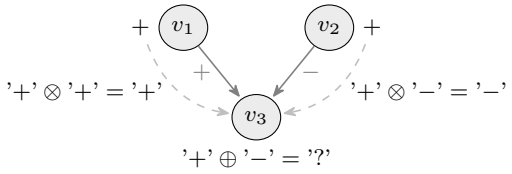

Fig. 4: An example for propagating the sign of v_1 to v_3. During the propagation, all other parents of v_3 (here only v_2) are also taken into account. Both v_1 and v_2 are assigned the sign '+' and therefore v_3 receives the two messages '+'⊗'+' = '+' and '+'⊗'−' = '−', which are then combined using the sign addition operator to obtain '+' ⊕ '−' = '?' as a new sign for v_3.

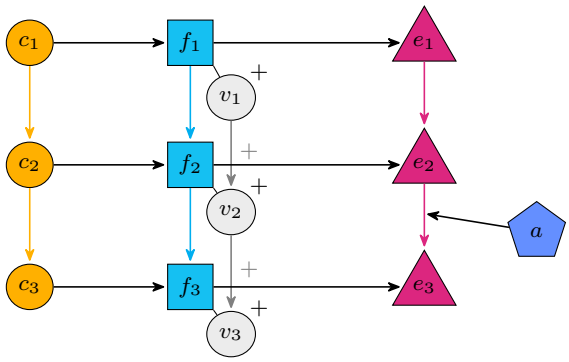

Fig. 5: A visualisation of an FMEA model with current state $s = \langle\{\text{tooHigh}\}, \{\text{tooHigh}\}, \{\text{tooHigh}\}\rangle$ (the variable assignments $v_1 = v_2 = v_3 = \text{tooHigh}$ are indicated by the '+' signs next to the variables). Action a is a preventive action for e_2, i.e., applying a sets $v_2 = \text{normal}$. The effect of a is then propagated according to the failure hierarchy such that the successor state after applying a in s is $s' = \langle\{\text{tooHigh}\}, \{\text{normal}\}, \{\text{normal}\}\rangle$.

ing that its value is "tooHigh" (i.e., the sign of v_2 is '+'), however, we cannot assign v_3 the result of the propagation from v_1 to v_3 (i.e., $'+' \otimes '+' = '+'$) because there is another influence of v_2 on v_3 (i.e., $'+' \otimes '-' = '-'$). Consequently, we know that v_3 both increases (due to v_1) and decreases (due to v_2) at the same time and we cannot infer whether v_3 will eventually become "tooLow" or "tooHigh" in this situation. The algorithm therefore uses sign addition on all messages from the parents of v_3, such that the new sign for v_3 is $'+' \oplus '-' = '?'$.

After the propagation of signs is finished, the new signs are returned and converted back to values in $\{\text{tooLow}, \text{normal}, \text{tooHigh}\}$ (in Algorithm 1). The new state s' obtained from the propagation is then added to the list of possible successor states. Performing qualitative causal reasoning instead of just setting the value of the variable v_i decreases the number of reachable states, i.e., states that contain inconsistent information cannot be reached and hence the state space is smaller than it would be without the propagation of information.

Example 9 For a more comprehensive example of the qualitative causal reasoning algorithm to compute possible successor states, take a look at the FMEA model depicted in Fig. 5. The FMEA model contains three failures $e_1 := \text{right\_critical}(v_1)$, $e_2 := \text{right\_critical}(v_2)$, and $e_3 := \text{right\_critical}(v_3)$ with failure hierarchy $e_1 \to e_2 \to e_3$, i.e., $(e_1, e_2) \in E2E$ and $(e_2, e_3) \in E2E$. It holds that $\text{range}(v_1) = \text{range}(v_2) = \text{range}(v_3) = \{\text{normal}, \text{tooHigh}\}$. Further, let a be a prevention action attached to (e_2, e_3) and $\mathcal{G} = (\{v_1, v_2, v_3\}, \{(v_1, v_2, +), (v_2, v_3, +)\})$ be the corresponding graph encoding the qualitative relationships between v_1, v_2, and v_3. When applying action a in state $s = \langle\{\text{tooHigh}\}, \{\text{tooHigh}\}, \{\text{tooHigh}\}\rangle$, the successor states are computed as follows. As a sets $v_2 = \text{normal}$, $\sigma = '0'$ is propagated starting from v_2. During initialisation in Line 5, the algorithm assigns $signs[v_1] = signs[v_2] = signs[v_3] = '+'$ (because $v_1 = v_2 = v_3 = \text{tooHigh}$ in s). In the first call of *propagate_rec*, it holds that $u = v = v_2$ and hence there are no parents of v (because the ingoing edges of v_2 have been removed before calling *propagate*). Hence, after initially setting $signs[v_2] = '0'$, the first update being made is $signs[v_2] = signs[v_2] \oplus \sigma = '0' \oplus '0' = '0'$. Then, v_2 is marked as visited and in the next call of *propagate_rec*, it holds that $u = v_2$, $v = v_3$, and $\sigma = signs[v_2] \otimes '+' = '0'$. As v_3 has no other parents apart from v_2, the next update being made after initially setting $signs[v_3] = '0'$ is $signs[v_3] = signs[v_3] \oplus \sigma = '0' \oplus '0' = '0'$. Afterwards, v_3 is marked as visited and the propagation algorithm terminates as there are no children for v_3. Finally, the signs of the variables are translated back to values in their range such that the new state is $s' = \langle\{\text{tooHigh}\}, \{\text{normal}\}, \{\text{normal}\}\rangle$. As $v_2 = \text{normal}$ is the only possible outcome of a, there are no other successor states apart from s'. Note that the value of v_1 is left unchanged as the prevention action performed on v_2 has no effect on v_1.

If additional information about quantitative relationships between variables is available, it is also possible to use quantitative causal reasoning [17] instead of qualitative causal reasoning. Before we show how solving the MDP of an FMEA model yields optimal therapies for patients in the medical domain, we note that it is also conceivable to employ a partially observable MDP [11,33] instead of an MDP to formalise the FMEA model. However, the transformation from FMEA model to partially observable MDP is not immediately clear and dealing with partially observable MDPs is far more complex than dealing with MDPs, which is a major drawback especially in the medical domain where acquiring the observation probabilities required by a partially observable MDP is a highly difficult task (as these probabilities are usually not known).

Next, we give a full algorithm taking as input an FMEA model and patient data of a specific patient to return an optimal therapy for that patient.

4 Automated Computation of Optimal Therapies in the Medical Domain

Solving the MDP from Section 3 yields an optimal policy π^* which maps every state to an optimal action. The optimal policy π^* can then be used to compute

Algorithm 3: Compute Optimal Therapy

```
1  function optimal_therapy(F, s_0, s_g, θ, D)
2      M ← fmea_to_mdp(F, s_0);
3      π* ← solve_mdp(M);
4      therapy ← [];
5      s' ← s_0;
6      repeat
7          s ← s';
8          a ← π*(s);
9          push(a, therapy);
10         s' ← a(s, D);
11     until s' = s_g ∨ R(s, a, s') > θ;
12     return therapy;
```

therapies for patients. Executing an action $a = \pi^*(s)$ in state s for an individual patient yields a specific successor state s' for which π^* also returns the best possible action to take. Thus, the optimal policy π^* directly corresponds to an optimal therapy. By adding a goal state to the MDP or a threshold on the reward, we can formulate an algorithm that computes the optimal therapy according to a given FMEA model.

Algorithm 3 outlines how to compute the optimal therapy for a specific patient according to a given FMEA model $\mathcal{F}$. The initial state s_0 is given by the available evidence for the patient. First, the algorithm transforms $\mathcal{F}$ into an MDP $\mathcal{M}$ and then solves $\mathcal{M}$ to obtain the optimal policy π^*. The algorithm then iteratively applies the optimal policy to the current state s until either the goal state s_g (it is also conceivable to have a set of goal states instead of a single goal state) is reached or the reward $R(s, a, s')$ reaches a user-defined threshold θ. All actions that have been applied as part of the optimal policy are appended to the resulting therapy. Whenever an action a is applied, the patient data D are taken into account to determine the unique successor state s' (the patient data contains the information about the exact result of the applied action, e.g., the outcome of a detection action).

Example 10 Take a look again the example shown in Fig. 2. The model states that too much interstitial fluid volume results in an interstitial pulmonary edema, i.e., $e_1 = \text{right\_critical}(v_1)$, and too little diffusing capacity of the lung impairs the gas exchange, i.e., $e_2 = \text{left\_critical}(v_2)$. In particular, it holds that $\text{range}(v_1) = \{\text{normal}, \text{tooHigh}\}$ and $\text{range}(v_2) = \{\text{normal}, \text{tooLow}\}$ and the edge $v_1 \overset{-}{\rightarrow} v_2$ implies that if v_1 increases, it causes v_2 to decrease, i.e., if the interstitial fluid volume is too high, the diffusing capacity of the lung will eventually become too low. Moreover, we have $pre = \{(p_1, v_1 = \text{tooHigh})\}$, i.e., the action p_1 can only be applied if an interstitial pulmonary edema is detected. For the sake of this example, let $\text{sev}(e_1) = 5$, $\text{occ}(e_1) = 4$, $\text{det}(e_1) = 9$, $\text{sev}(e_2) = 7$, $\text{occ}(e_2) = 5$, $\text{det}(e_2) = 9$, $p_{e_1} = 0.4$, and $p_{e_2} = 0.5$. The corresponding MDP to this FMEA model consists of the action space $A = \{d_1, p_1\}$ and the state space $S = \{s_1, \ldots, s_9\}$, where

- $s_1 = (\{\text{normal}\}, \{\text{normal}\})$,
- $s_2 = (\{\text{normal}\}, \{\text{tooLow}\})$,
- $s_3 = (\{\text{tooHigh}\}, \{\text{normal}\})$,
- $s_4 = (\{\text{tooHigh}\}, \{\text{tooLow}\})$,
- $s_5 = (\{\text{normal}, \text{tooHigh}\}, \{\text{normal}\})$,
- $s_6 = (\{\text{normal}, \text{tooHigh}\}, \{\text{tooLow}\})$,
- $s_7 = (\{\text{normal}\}, \{\text{normal}, \text{tooLow}\})$,
- $s_8 = (\{\text{tooHigh}\}, \{\text{normal}, \text{tooLow}\})$, and
- $s_9 = (\{\text{normal}, \text{tooHigh}\}, \{\text{normal}, \text{tooLow}\})$.

The initial state for a patient without evidence is s_9 and the goal state in this example is s_1. Solving the MDP corresponding to the given FMEA model then yields a policy π^* that returns appropriate actions for each state, e.g., $\pi^*(s_4) = p_1$ such that the goal state is reached immediately after applying p_1 in s_4 (due to the propagation of the effect of p_1). If we consider a patient arriving at a hospital who has an interstitial pulmonary edema and thus an impaired gas exchange (but this diagnosis is not known beforehand), the optimal therapy computed by Algorithm 3 would be $\langle d_1, p_1 \rangle$ (because the patient data D tells us that applying d_1 in state s_9 results in a transition to state s_4). In other words, the recommended therapy would be to first apply the detection action d_1 (which then finds out about the patient's interstitial pulmonary edema) and afterwards to apply the prevention action p_1 (whose preconditions are then satisfied) to treat the disease accordingly.

Before we conclude this paper, we discuss further applications and limitations of our proposed approach.

5 Discussion

The general approach of transforming an FMEA model into an MDP to automatically compute the best sequence of actions to reduce the risk as much as possible is obviously not restricted to the medical domain. Hence, industries such as the automotive industry, the aerospace industry, and manufacturing industries in general, which commonly apply the FMEA approach, can also vastly benefit from the automatic planning and acting capabilities provided by the MDP.

However, the presented approach to transform an FMEA model into an MDP clearly has its own limitations and can be further refined, e.g., by integrating different ranges of variables and hence additional failures having semantics that are different from $\text{left\_critical}(v_i)$ and $\text{right\_critical}(v_i)$, by adding costs to actions, by

handling erroneous measurements obtained from detection actions, and so on. It is also possible to incorporate a probability distribution over the variables in V to allow for probabilistic (quantitative) causal inference [17] instead of merely using qualitative causal inference. Obtaining more fine-grained FMEA models, however, is a serious challenge in the medical domain—and most likely also in other domains—as obtaining the additionally required information involves a lot of effort. Another limitation of the MDP is its scalability. As we have seen in Example 10, the state space of the MDP becomes quite large even for a small FMEA model. Even though the propagation of action effects during the computation of successor states results in many states not being reachable at all (and hence they could be omitted from the MDP after an initial reachability check), the size of the state space is still a limitation when it comes to solving the MDP for large FMEA models. To encounter the scalability problem induced by large state spaces, reinforcement learning [26] might be applied as a remedy. Similar to the AlphaZero program [23] (which has been developed to master the games of chess, shogi, and go where the state spaces are also huge), one could use reinforcement learning to learn an approximation of the optimal policy. The idea is to sample an initial state and random sequences of actions for which then the reward of the resulting state is used as a measure of quality for that particular action sequence. By repeating the sampling procedure for various initial states and action sequences, an approximation of the optimal policy can be obtained. The development of such a reinforcement learning approach, however, is out of the scope of this paper and hence an interesting direction for future work.

Before we conclude this paper, we give an outlook on possibilities to augment large language models (LLMs) with formalised domain knowledge represented in formal models such as FMEA models and their corresponding MDP. We believe that formal models can be used to generate training data for the fine-tuning step of an LLM by sampling the model. The MDP allows us to generate training data for LLMs by computing therapies for a lot of different initial states, thresholds, and patient data. Given the generated data, the computed therapies can be verbalised (i.e., translated to natural language) and afterwards, the verbalised data can be used to fine-tune a pre-trained LLM. By integrating the knowledge of domain experts represented in the formal model into the LLM, the LLM might produce less hallucinations. Furthermore, it is conceivable to use a formal model again to validate the output of the LLM by translating the output of the LLM into the syntax of the formal model and then using the formal model to check whether the input-output pair of the LLM matches the computed optimal therapy.

6 Conclusion

We present a formal framework to conduct automated planning and acting in FMEA models. In particular, we apply FMEA to the medical domain and transform the resulting FMEA model into an MDP to automatically compute optimal therapies for individual patients. Further, we introduce qualitative causal reasoning to compute the successor states in the MDP after applying an action, yielding a fully automated algorithm to compute a therapy for a particular patient.

Future work includes the application of reinforcement learning to encounter the state space explosion of the MDP, as well as the augmentation of general LLMs with formalised domain knowledge.

Acknowledgements

This work is funded by the Medical Cause and Effects Analysis (MCEA) project. The authors also thank the anonymous reviewers for their insightful comments. This version of the article has been accepted for publication, after peer review, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/s13218-023-00810-z.

References

1. Adachi, W., Lodolce, A.E.: Use of Failure Mode and Effects Analysis in Improving the Safety of i.v. Drug Administration. American Journal of Health-System Pharmacy **62**, 917–920 (2005)
2. Akbulut, F.P., Akkur, E., Akan, A., Yarman, B.S.: A Decision Support System to Determine Optimal Ventilator Settings. BMC Medical Informatics and Decision Making **14**, 3 (2014)
3. Battefeld, D., Kopp, S.: Formalizing Cognitive Biases in Medical Diagnostic Reasoning. In: Proceedings of the Eighth Workshop on Formal and Cognitive Reasoning (FCR-22). pp. 102–118. CEUR (2022)
4. Bellman, R.: A Markovian Decision Process. Journal of Mathematics and Mechanics **6**, 679–684 (1957)
5. Berner, E.S.: Clinical Decision Support Systems. Springer (2007)
6. Bottino, D.A., Giannella-Neto, A., David, C.M.N., Melo, M.F.V.: Decision Support System to Assist Mechanical Ventilation in the Adult Respiratory Distress Syndrome. International Journal of Clinical Monitoring and Computing **14**, 73–81 (1997)
7. Druzdzel, M.J., Henrion, M.: Efficient Reasoning in Qualitative Probabilistic Networks. In: Proceedings of the Eleventh National Conference on Artificial Intelligence. pp. 548–553. AAAI Press (1993)

8. García Aguirre, P.A., Pérez-Domínguez, L., Luviano-Cruz, D., Solano Noriega, J.J., Martínez Gómez, E., Callejas-Cuervo, M.: PFDA-FMEA, an Integrated Method Improving FMEA Assessment in Product Design. Applied Sciences **11**, 1406 (2021)
9. Hartmann, S.M., Farris, R.W., Yanay, O., DiBlasi, R.M., Kearney, C.N., Zimmerman, J.D., Carlin, K., Zimmerman, J.J.: Interaction of Critical Care Practitioners with a Decision Support Tool for Weaning Mechanical Ventilation in Children. Respiratory Care **65**, 333–340 (2020)
10. Hauskrecht, M., Fraser, H.: Planning Treatment of Ischemic Heart Disease with Partially Observable Markov Decision Processes. Artificial Intelligence in Medicine **18**, 221–244 (2000)
11. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and Acting in Partially Observable Stochastic Domains. Artificial Intelligence **101**, 99–134 (1998)
12. Karbing, D.S., Spadaro, S., Dey, N., Ragazzi, R., Marangoni, E., Dalla Corte, F., Moro, F., Lodahl, D., Hansen, N.S., Winding, R., Rees, S.E., Volta, C.A.: An Open-Loop Physiologic Model-Based Decision Support System Can Provide Appropriate Ventilator Settings. Critical Care Medicine **46**, 642–648 (2018)
13. Kisling, K., Johnson, J.L., Simonds, H., Zhang, L., Jhingran, A., Beadle, B.M., Burger, H., du Toit, M., Joubert, N., Makufa, R., Shaw, W., Trauernicht, C., Balter, P., Howell, R.M., Schmeler, K., Court, L.: A Risk Assessment of Automated Treatment Planning and Recommendations for Clinical Deployment. Medical Physics **46**, 2567–2574 (2019)
14. Liu, L., Shuai, M., Wang, Z., Li, P.: Use-Related Risk Analysis for Medical Devices Based on Improved FMEA. Work **41**, 5860–5865 (2012)
15. Najafpour, Z., Hasoumi, M., Behzadi, A., Mohamadi, E., Jafary, M., Saeedi, M.: Preventing Blood Transfusion Failures: FMEA, an Effective Assessment Method. BMC Health Services Research **17**, 1–9 (2017)
16. Pearl, J.: Causal Diagrams for Empirical Research. Biometrika **82**, 669–688 (1995)
17. Pearl, J.: Causality: Models, Reasoning and Inference. Cambridge University Press, 2nd edn. (2009)
18. Pelletier, J.H., Horvat, C.M.: Can Computer Decision Support Help Us Follow Our Own Rules in Pediatric Acute Respiratory Distress Syndrome? Pediatric Critical Care Medicine: A Journal of the Society of Critical Care Medicine and the World Federation of Pediatric Intensive and Critical Care Societies **21**, 1000–1001 (2020)
19. Press, D.: Guidelines for Failure Mode and Effects Analysis (FMEA), for Automotive, Aerospace, and General Manufacturing Industries. CRC Press (2003)
20. Press, D.: Guidelines for Failure Modes and Effects Analysis for Medical Devices. CRC Press (2018)
21. Rudowski, R., East, T.D., Gardner, R.M.: Current Status of Mechanical Ventilation Decision Support Systems: A Review. International Journal of Clinical Monitoring and Computing **13**, 157–166 (1996)
22. Segismundo, A., Miguel, P.A.C.: Failure Mode and Effects Analysis (FMEA) in the Context of Risk Management in New Product Development: A Case Study in an Automotive Company. International Journal of Quality & Reliability Management **25**, 899–912 (2008)
23. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play. Science **362**, 1140–1144 (2018)
24. Stamatis, D.H.: Failure Mode and Effect Analysis: FMEA from Theory to Execution. Quality Press (2003)
25. Sutton, R.T., Pincock, D., Baumgart, D.C., Sadowski, D.C., Fedorak, R.N., Kroeker, K.I.: An Overview of Clinical Decision Support Systems: Benefits, Risks, and Strategies for Success. NPJ digital medicine **3**, 17 (2020)
26. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (2018)
27. Sward, K.A., Newth, C.J.L.: Computerized Decision Support Systems for Mechanical Ventilation in Children. Journal of Pediatric Intensive Care **5**, 95–100 (2016)
28. van Baalen, S., Boon, M., Verhoef, P.: From Clinical Decision Support to Clinical Reasoning Support Systems. Journal of Evaluation in Clinical Practice **27**, 520–528 (2021)
29. Viscariello, N., Evans, S., Parker, S., Schofield, D., Miller, B., Gardner, S., Fong de los Santos, L., Hallemeier, C., Jordan, L., Kim, E., Ford, E.: A Multi-Institutional Assessment of COVID-19-Related Risk in Radiation Oncology. Radiotherapy and Oncology **153**, 296–302 (2020)
30. Wellman, M.P.: Fundamental Concepts of Qualitative Probabilistic Networks. Artificial Intelligence **44**, 257–303 (1990)
31. Yazdi, M., Daneshvar, S., Setareh, H.: An Extension to Fuzzy Developed Failure Mode and Effects Analysis (FDFMEA) Application for Aircraft Landing System. Safety Science **98**, 113–123 (2017)
32. Zhang, W., Wang, H.: Diagnostic Policies Optimization for Chronic Diseases Based on POMDP Model. Healthcare **10**, 283 (2022)
33. Åström, K.J.: Optimal Control of Markov Processes with Incomplete State Information. Journal of Mathematical Analysis and Applications **10**, 174–205 (1965)