Explanation of Black Box AI for GDPR related Privacy using Isabelle

Florian Kammüller

Middlesex University London and
Technische Universität Berlin
f.kammueller@mdx.ac.uk

Abstract. In this paper, we present a methodology for constructing explanations for AI classification algorithms. The methodology consists of constructing a model of the context of the application in the Isabelle Infrastructure framework (IIIf) and an algorithm that allows to extract a precise logical rule that specifies the behaviour of the black box algorithm thus allowing to explain it. The explanation is given within the rich logical model of the IIIf. It is thus suitable for human audiences. We illustrate this and validate the methodology on the application example of credit card scoring with special relation to the right of explanation as given by the GDPR.

1 Introduction

Artificial intelligence (AI) uses various methods of machine learning to solve problems automatically. Some of the used methods, for example linear regression or decision trees are amenable to human understanding. However, other very successful ones, for example deep learning methods and convolutional neural networks (CNN), are black boxes for humans: it is not clear from the outside how the machine intelligence arrives at decisions. In critical applications, however, it is absolutely necessary that humans can understand what is going on and why.

We provide a method for explanation using the expressive Higher Order logic of the interactive theorem prover Isabelle. The Isabelle Infrastructure framework (IIIf) provides rich contexts for actors, infrastructures and policies to enable explanations of black box machine learning decisions to humans. This can be particularly relevant for privacy critical application. Black box algorithms are trained on data sets whose classification may have human biases but those are hidden in the opaqueness of the learning algorithm. Explanation is necessary to shed light into this. We propose a process of precondition refinement to arrive at logical rules for explanation using counterfactuals for iterating the refinement process. We illustrate and validate the proposed methodology by an example of credit scoring. There private information is used in the automated decision process of credit scoring guided by a black box AI algorithm. The GDPR grants a right of explanation. We show how our approach using attack trees and an expressive logical rule for explanation serves the main purposes of GDPR explanation. All Isabelle sources are available [14].

2 Background and Related Work

2.1 Explanation

Bau et al's article "Explaining Explanation [...]" [5] gives a good overview of the techniques used for explainable AI (XAI). The more recent work [1] provides a critical analysis of current literature on the field of XAI providing some challenges primarily featuring the post-hoc explanation of black box machine learning like CNN and Deep Learning and providing human comprehensible explanations. Belle and Papantonis [2] also give a very comprehensive survey of current explanation approaches including very accessible illustrations of their use on human centric examples.

Pieters distinguishes the main incentives of explanation as transparency (for the user) and justification [23]. Explanation trees may be used to visualize the relation of explanation goals and their subgoals according to Pieters providing "a tree in which the goals and subgoals of an explanation are ordered systematically" [23]. This work is a very strong motivation to our approach to explainability because explanation trees have much in common with attack trees. An attack tree makes an attack more transparent by a step by step process that can be characterized as "attack tree refinement" [7]. Ultimately, the attack tree refinement leads to a fully expanded explanation that can be automatically verified on the model as is shown in the Isabelle Insider framework [8,10]. Thus similar to an explanation tree, a sub-tree of an attack tree "explains" the attack expressed by the parent node.

2.2 GDPR, Explanation and IIIf

The GDPR explicitly mentions a right of explanation. Wachter et al [25] investigate the use of counterfactuals for the explanation of automated decisions. They argue that counterfactuals are in themselves sufficient to provide explanations. We challenge their approach (see Section 5.1). Nevertheless, we also adopt the use of counterfactuals but only as a means to construct a general explanation as a logical rule. However, Wachter et al also give a very detailed analysis of where explanation is mentioned in the GDPR [25]. They clearly identify the purposes of an individual who would want to claim an explanation against a data controller based on the GDPR. Explanations serve three main purposes [25, Section 5].

- *Understand decisions*: provide transparency of the scope of automated decisions and reasons.
- *Contest decisions*: provide the means to challenge a decision.
- *Alter future decisions*: provide help to adapt future behaviour to receive the preferred outcome.

We will show that the logical method of explanation we provide serves all three purposes (see Section 4.3).

In terms of logical modeling and analysis, it is worth mentioning that the IIIf has been used for GDPR relevant applications. In fact, [9] evaluates IoT scenarios

with respect to GDPR related privacy. This work shows that the IIIf can be applied to provide *privacy by design* – one of the major principles stipulated by the GDPR. However, this early application of IIIf in the context of the GDPR can be seen as a formal experiment inspired by the concepts propagated by the GDPR and to advocate the use of formal verification to support GDPR compliance checking of IoT architectures. What the current work achieves is much closer to actual applications of the GDPR in law practice. It is thus more relevant to the application of the GDPR as a law to privacy related societal issues. This is illustrated as well by the case study we present in this paper where the explanation that our logical method IIIf provides can serve as a basis for challenging a decision made by a data controller using an AI based black box decision algorithm.

2.3 Isabelle Infrastructure framework (IIIf)

Attack trees are fully embedded as "first-class citizens" into the logic in the Isabelle Insider and Infrastructure framework (IIIf). It is thus possible to provide a formal semantics for valid attacks based on Kripke structures and the temporal logic CTL as well as to derive an efficient decision procedure. Code is generated in the programming languages Scala for deciding the validity of attack trees.

The Isabelle Infrastructure framework (IIIf) is implemented as an instance of Higher Order Logic in the interactive generic theorem prover Isabelle/HOL [22]. The framework enables formalizing and proving of systems with physical and logical components, actors and policies. It has been designed for the analysis of insider threats. However, the implemented theory of the temporal logic CTL combined with Kripke structures and its generic notion of state transitions are a perfect match to be combined with attack trees into a process for formal security engineering [4] including an accompanying framework [11]. In the current paper, we show that the IIIf can also be used for explaining AI decisions made by black box algorithms. We provide here a very brief overview of the main features of the IIIf.

Kripke structures, CTL and Attack Trees A number of case studies have contributed to shape the Isabelle framework into a general framework for the state-based security analysis of infrastructures with policies and actors. Temporal logic and Kripke structures are deeply embedded into Isabelle's Higher Order logic thereby enabling meta-theoretical proofs about the foundations: for example, equivalence between attack trees and CTL statements have been established [8] providing sound foundations for applications. This foundation provides a generic notion of state transition on which attack trees and temporal logic can be used to express properties for applications. The logical concepts and related notions thus provided for sound application modeling are:

– Kripke structures and state transitions:
  A generic state transition relation is $\rightarrow$; Kripke structures over a set of states

`t` reachable by $\rightarrow$ from an initial state set `I` can be constructed by the `Kripke` constructor as

```
Kripke {t. ∃ i ∈ I. i →* t} I
```

– CTL statements:
We can use the Computation Tree Logic (CTL) to specify dependability properties as

```
K ⊢ EF s
```

This formula states that in Kripke structure `K` there is a path (`E`) on which the property `s` (given as the set of states in which the property is true) will eventually (`F`) hold.

– Attack trees:
attack trees are defined as a recursive datatype in Isabelle having three constructors: $\oplus_\vee$ creates or-trees and $\oplus_\wedge$ creates and-trees. And-attack trees $l \oplus^s_\wedge$ and or-attack trees $l \oplus^s_\vee$ consist of a list of sub-attacks which are themselves recursively given as attack trees. The third constructor takes as input a pair of state sets constructing a base attack step between two state sets. For example, for the sets `I` and `s` this is written as $\mathcal{N}_{(\mathtt{I},\mathtt{s})}$. As a further example, a two step and-attack leading from state set `I` via `si` to `s` is expressed as

$$\vdash \ [\mathcal{N}_{(\mathtt{I},\mathtt{si})},\mathcal{N}_{(\mathtt{si},\mathtt{s})}] \oplus_\wedge^{(\mathtt{I},\mathtt{s})}$$

– Attack tree refinement, validity and adequacy:
Attack trees can be constructed also by a refinement process but this differs from the system refinement presented in the paper [12]. An abstract attack tree may be refined by spelling out the attack steps until a valid attack is reached:
`⊢A :: (σ:: state) attree)`.
The validity is defined constructively so that code can be generated from it. Adequacy with respect to a formal semantics in CTL is proved and can be used to facilitate actual application verification.

The IIIf has a wide range of applications ranging from Insider threats in auction protocols [17] and airplane policies [16], security and privacy of IoT healthcare systems, for example, [9,11], the Quantum Key Distribution protocol [10], Inter-blockchain protocols [19], the Corona-Warn App [18,12], and awareness in social networks and unintentional insiders [15] .

The potentials of using the IIIf for explanation (for AI and security) has already been presented in an earlier paper but at the level of position paper only [13] while the current paper provides a feasible methodology, an implementation of explanation within the IIIf illustrating it on an application to a relevant case study in the context of GDPR related privacy. Online sources are available [14].

3 Case Study of Credit Scoring

In this section, we give a brief introduction to credit scoring and its relevant factors to motivate the case study that is used to illustrate how IIIf is applied to it to provide a basis for explanation.

3.1 Credit scoring

Credit scores are ranks assigned to people to allow quantifying their "financial fitness" [3]. These scores are used by banks as well as credit institutes to decide whether a client may receive a credit card or more importantly whether a potential lender may receive a mortgage. It also influences the interest rate you may receive which can lead to disadvantaging poorer people. According to Internet publications [3,6] the research supporting the actual credit scoring "has come from ClearScore.com"[3]. An interactive map provided by this company allows to easily check credit scores as is illustrated in Figure 1. An open question is

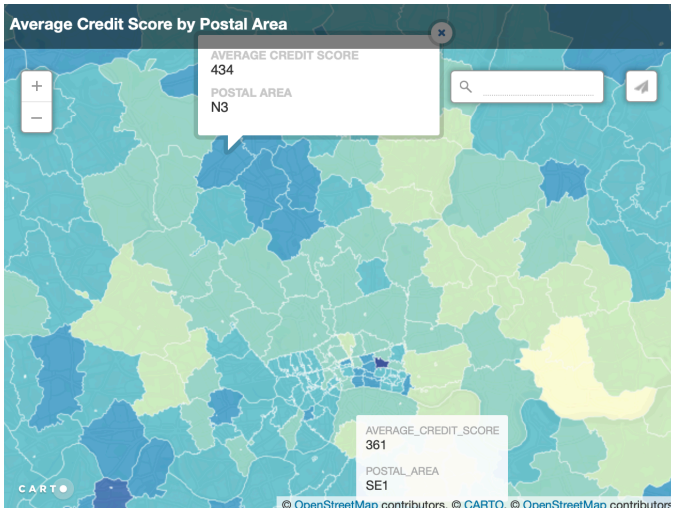

Fig. 1. Interactive map illustrates how credit scores differ in London districts [6].

how such credit scores are created as they rely on private data. As Bull writes [3]: "A higher income does not automatically lead to a higher credit score. For example, residents in Kensington and Chelsea are among the capital's highest average earners at £131,000 a year but they rank in the middle of the credit score table." Nevertheless, it is quite obvious that such scores are used by credit institutes. It seems rather likely that also AI based decision making procedures are applied within financial institutions. In order to clarify such opaque relations logical modeling can help as it remodels the actual context of the original data collection and thus may show up any biases used. In the next section, we present a simple example to illustrate how the credit scoring scenario can be represented as an infrastructure model in IIIf.

3.2 Model in IIIf

The Isabelle Infrastructure framework supports the representation of infrastructures as graphs with actors and policies attached to nodes. These infrastructures are the *states* of a Kripke structure describing the credit scoring scenario. The behaviour is defined by a transition relation on states. This transition between states is triggered by non-parameterized actions `put`, `eval`, `move`, and `get` executed by actors. Actors are given by an abstract type `actor` and a function `Actor` that creates elements of that type from identities (of type `string` written `''s''` in Isabelle). Actors reside at locations of an infrastructure graph of type `igraph` constructed by its constructor `Lgraph`.

```
datatype igraph =
        Lgraph (location × location)set
                location ⇒ identity set
                identity ⇒ (dlm × data) set
                data ⇒ bool
                (identity × bool option)set
```

For the current application to the credit scoring scenario, this graph contains the actual location graph of type `(location × location)set` given by a set of location pairs, and a function of type `location ⇒ identity set` that assigns the actors to their current location. The third component of the datatype `igraph` is of type `identity ⇒(dlm ×data) set`. It assigns security labeled data to actors. The label type is called `dlm` as a reference to the decentralized label model by Myers and Liskov who inspired it [21]. It is a pair of type `actor ×actor set` defining the owner and the readers of a `data` item. The type `data` contains the private data of users. For our example, we use the location, the salary, their date of birth and ethnicity. [1]

```
data = location × nat × dob × ethnicity
```

The fourth component of type `data ⇒ bool` is the black box function: effectively a table that contains the credit approval decision for given data inputs. The final component of type `(identity ×bool option)set` records that a user has requested a credit approval by uploading their identity together with a boolean field that contains the future decision of the credit approval to the set of `requests`. The second boolean component containing the answer is lifted by the option type constructor that enables an undefined value `None` to flag that there has not been any response yet.

Each of the components of the type constructor is equipped with a corresponding projection function that allows to access this component in an instance of this type constructor (an element of this type). These projection functions are named `gra` for the set of pairs of location representing the infrastructure graph, `agra` for the assignment of actors to locations, `dgra` for the data at that location, `bb` for the black box, and `requests` for the pairs of request and approval decision of actors of requesting credits.

[1] The latter two type definitions are omitted for brevity.

We omit some standard constructions for infrastructure assembly and the policy definition from local policies (see for example [12]). A generic state transition relation over Kripke structures is defined together with logic and decision procedures for IIIf. This is then instantiated to concrete applications of the IIIf – like in the current credit scoring example – by defining the rules for the state transition relation over a defined infrastructure type – as given by the above `igraph`. This state transition relation then implements the behaviour for credit scoring systems by explaining how actions executed by actors change the infrastructure state. The execution of actions is conditional on enabledness as defined by the local policies and other conditions of the context. For credit scoring systems, we consider here the actions `put` representing that a client requests a credit approval and `eval` where an entitled client (presumably a credit institute) executes a requested credit application.

In the precondition of the rule for a `put` action, the actor `a` residing at location `l` in the infrastructure graph `G` (given by the predicate `Actor a` $@_G$ `l`) who is enabled to put a request, uploads their data to the `requests G` field into the infrastructure graph `G`. A potential credit institute `Actor c` can see a new request since now there is a new pair `(a, None)` in the requests set where the second component of this pair is flagged by the `None` constructor of the option type as "unprocessed" while the first element is the requesting actor's identity `a`.

```
put: G = graphI I ⟹ Actor a @_G l ⟹ Actor c ∈ actors G ⟹
  enables I l (Actor a) put ⟹
  I' = Infrastructure
         (Lgraph (gra G)(agra G)(dgra G)(bb G)
                 (insert (a, None)(requests G)))
         (delta I)
⟹ I → I'
```

The action `eval` allows evaluation of a request filed by actor `a` by a (presumable) credit institute `c` given that `c` is is contained in the readers set of the `dlm` label `lb` that is given as the second element of the first element of the data item `dgra G a`. Also `c` needs to be enabled to evaluate requests by the local policy. Given these prerequisites, the actual evaluation is done by applying the black box function to the data item `d` and recording the outcome in the second component of the corresponding pair for `a` in the `requests` set.

```
eval: G = graphI I ⟹ Actor a @_G l ⟹  l ∈ nodes G ⟹
        Actor c ∈ actors G ⟹  (a, None) ∈ requests G ⟹
        (lb,d) = dgra G a ⟹ Actor c ∈ snd lb ⟹
        enables I l (Actor c) eval ⟹
  I' = Infrastructure
         (Lgraph (gra G)(agra G)(dgra G)(bb G)
                 (insert (a, Some((bb G) d))(requests G - {(a, None)}))
         (delta I)
⟹ I → I'
```

We omit the state transition rules for the actions `get` and `move`. They will be illustrated in the evaluation of the example below. For details of their definitions see the online sources [14].

The above infrastructure Kripke model for credit scoring formalises credit scoring scenarios enabling reasoning in general about all instances. To simulate concrete example scenarios, we can use the generic nature of the IIIf with its polymorphic Kripke structure and state transition. Defining a locale [20] named `CreditScoring` allows fixing some concrete values for the actors, locations, and local policies and inherits all general definitions and properties of infrastructures from the framework. For simplicity we consider just two actors Alice and Bob and a credit institute `CI`.

```
locale CreditScoring =
defines CreditScoring_actors = {''Alice'', ''Bob'', ''CI''}
```

The locale allows to initialize a concrete `igraph` with these and other values. Moreover it serves to illustrate the explanation process that we are going to present next.

4 Precondition Refinement Process (PCR Cycle)

In this section, we define a Precondition Refinement process (PCR cycle) which is a cyclic method to derive a general logical characterization of what the black box mechanism decides within any given state of the world. A possible world is described in the IIIf as a Kripke state comprising actors, policies and infrastructures including any features necessary to specify the context of a human centric scenario. After defining the process, we continue by illustrating its use on the previously introduced credit scoring system.

4.1 Definition of PCR cycle

In contrast to the RR-cycle [12], we do not refine a system specification instead we refine the precondition of an explanation rule using the dynamic behaviour of an infrastructure system. But similar to the RR-cycle, we use attack trees to find "failure states", that is, states in which a desirable outcome is not given. These failure states allow us to find counterfactuals, which are local rules for specific instances for which the desirable outcome is achieved. The preconditions of these local rules guide the refinement of a general precondition. The refinement of the precondition is repeated until it yields a general rule for explanation. The starting point for this cyclic process of precondition refinement is an attack tree, i.e. a proof of a temporal property of the form $\mathtt{M} \vdash \mathsf{EF}\ \neg DO$ showing that "failure" states in the model `M` can be reached that do not fulfill the *desirable outcome (DO)*. This DO is comparable to what the "global policy" is in the RR-cycle[12]. The failure state can be used to define a *counterfactual*, essentially given as an additional precondition that would have provided an alternative path to a state

fulfilling the DO property. Besides helping to guide the refinement by counterfactuals, the DO also provides the termination condition of the cyclic precondition refinement process. Since the DO property is the positive classification of the AI algorithm given as a black box, the process yields a general explanation rule that gives a precise logical description how the DO property can be achieved.

This is in a nutshell the working of the PCR cycle. In what follows we provide its high level yet detailed algorithmic description including the formal definitions of the core concepts used. However, before we come to that we need to introduce how we formalise counterfactuals.

Counterfactuals A counterfactual is best explained by example. We give one that fits into the context of our case study: "if the client would have a monthly salary of 40K, he would have got the loan approval". Intuitively, counterfactuals try to illustrate facts in the current state of the world by showing alternative hypothetical developments of the world that feature the opposite case of the fact. It is not a coincidence that our explicit world model of Kripke structures and state transitions lends itself so naturally to modeling counterfactuals.

However, apart from modeling the different possible worlds and their evolution, we also need a metric on them. As Wachter observes "the concept of the "closest possible world" or the smallest change to the world that can be made to obtain a desirable outcome, is key throughout the discussion of counterfactuals" [25]. We use the step-relation between possible states (worlds) to define a unique notion of "closest" between three states. Intuitively, it formalises the closest predecessor `s` of two possible states `s'` and `s''` by stipulating that any other state $\mathtt{s}_0$ that is also a predecessor (with respect to $\rightarrow^*$) to states `s'` and `s''` must already be a predecessor to `s`.

Definition 1 (Closest State).

$$\mathtt{closest\ s\ s'\ s''} \equiv \mathtt{s} \rightarrow^* \mathtt{s'} \wedge \mathtt{s} \rightarrow^* \mathtt{s''} \wedge \forall\, \mathtt{s}_0.\ \mathtt{s}_0 \rightarrow^* \mathtt{s'} \wedge \mathtt{s}_0 \rightarrow^* \mathtt{s'} \Rightarrow \mathtt{s}_0 \rightarrow^* \mathtt{s}$$

This definition is used for defining counterfactuals with respect to a desirable outcome `DO` by simply stating that for a state `s` with $\neg$`DO s` there must be an alternative trace leading to another possible world `s''` with `DO s''` such that they are connected by a closest state `s'`. Using the definition of closest we can define this simply as the set of states for which such a closest predecessor exists.

Definition 2 (Counterfactuals). *Counterfactuals for a state `s` wrt a desirable property `DO` are the set of states `s''` with a closest predecessor `s'`.*

$$\mathtt{counterfactuals\ s\ DO} \equiv \{\mathtt{s''}.\ \mathtt{DO\ s''} \wedge (\exists\, \mathtt{s'}.\ (\mathtt{s'} \rightarrow^* \mathtt{s''}) \wedge \mathtt{closest\ s\ s'\ s''})\}$$

We will see the application of these concepts in the following algorithm.

PCR cycle algorithm

1. Using attack tree analysis in the CTL logic of the IIIf we *find the initial starting condition of the PCR*. The variable `B` is an element of a datatype for which we seek explanation (in the example it is actors).

   $\texttt{M} \vdash \mathsf{EF}\ \{\ \texttt{s} \in \texttt{states M.}\ \neg\ \texttt{DO(B, s)}\ \}.$

   This formula states that there exists a path (EF) on which eventually (F) a state `s` will be reached in which the desirable outcome is not true for `B`. The path corresponds to an attack tree (by adequacy [8]) designating failure states `s`.
2. *Find the (initial or refined) precondition using a counterfactual.*
   That is, for a state `s` in the set of failure states identified in the previous step
   (a) Find states `s'` and `s''` such that `closest s s' s''`, that is, $\texttt{s'} \rightarrow^* \texttt{s}$ and $\texttt{s'} \rightarrow^* \texttt{s''}$. In addition, `DO(B,s'')` must hold.
   (b) Identify the precondition $\texttt{pc}_i$ leading to the state `s''` where `DO` holds, that is, find an additional predicate Δ_i with $\Delta_i\texttt{(B, s')}$ and use it to extend the previous predicate $\texttt{pc}_i$ to $\texttt{pc}_{i+1}\texttt{:=}\ \texttt{pc}_i \wedge \Delta_i$.
3. *Generalisation.*
   Use again attack tree analysis in the CTL logic of the IIIf to check whether the following formula is true on the entire datatype of `B`: it is globally true (AG) that if the precondition $\texttt{pc}_i$ holds, there is a path on which eventually (EF) the desirable outcome `DO` holds.[2]

   $\forall\ \texttt{A. M} \vdash\ \mathsf{AG}\ \{\texttt{s} \in \texttt{states M.}\ \texttt{pc}_i\ \texttt{(A, s)} \longrightarrow\ \mathsf{EF}\ \{\texttt{s. DO(A, s)}\}\}$

   (a) If the check is negative, we get an attack tree, that is, IIIf provides an explanation tree for
   $\texttt{M} \vdash\ \mathsf{EF}\ \{\ \texttt{s} \in \texttt{states M.}\ \texttt{pc}_i\texttt{(A,s)} \wedge \neg\ \texttt{DO(A, s)}\ \}$
   and a set of failure states `s` with $\texttt{pc}_i\texttt{(A,s)}$ and the desirable outcome is not true: $\neg\texttt{DO(A,s)}$.
   In this case, *go to step 2. and repeat* with the new set of failure states in order to find new counterfactuals and refine the predicate.
   (b) If the check is positive, we have *reached the termination condition* yielding a precondition $\texttt{pc}_n$ such that for all `A`:
   $\texttt{M} \vdash\ \mathsf{AG}\ \{\ \texttt{s} \in \texttt{states M.}\ \texttt{pc}_n\ \texttt{(A, s)} \longrightarrow\ \mathsf{EF}\ \{\texttt{s. DO(A, s)}\}\ \}$

Note, that the analysis in Step 3 might potentially reveal a new variable as part of Δ_i over another datatype (locations in the example). This is not a problem as it will eventually lead to tease out the entire set of parameters that the black box decision procedure uses. We did not attempt to formalise it explicitly into the above algorithm description to keep the exposition easier understandable.

[2] Note, that the interleaving of the CTL-operators AG and EF with logical operators, like implication $\longrightarrow$ is only possible since we use a Higher Order logic embedding of CTL.

4.2 Applying the PCR cycle to credit scoring case study

We now demonstrate the PCR algorithm on our case study introduced in Section 3. We consider the scenario, where Bob gets an evaluation by the credit institute `CI` and it is not approved. Bob wants to understand why this is the case and requests an explanation. The experts in the credit institute cannot give this explanation as they have used a black box machine learning system `bb`. Now, the IIIf and the PCR algorithm can be used by modeling the scenario and using the `bb` system as a black box, that is, requesting only its classification output (verdict) for any given inputs.[3] The desirable outcome DO in an infrastructure state `s` is given by the pair that `a` filed having a `True` as second component (lifted by `Some`).

```
DO(a,s) ≡ (a, Some(True)) ∈ requests s
```

We show the run of the algorithm by going through its steps 1..3 for the application additionally ornating the numbers with $\alpha, \beta, \ldots$ to indicate the round of the algorithm.

α.1 For actor Bob, we use CTL modelchecking in the IIIf to verify the formula

```
Credit_Kripke ⊢
      EF { s ∈ states Credit_Kripke. ¬ DO(''Bob'', s) }.
```

From this proof, the IIIf allows applying Completeness and Correctness results of CTL [8] to derive the following attack tree.

$\vdash$ $[\mathcal{N}_{(\mathtt{I},\mathtt{C})}, \mathcal{N}_{(\mathtt{C},\mathtt{CC})}]\oplus_{\wedge}^{(\mathtt{I},\mathtt{CC})}$

The attack tree corresponds to a path leading from the initial state `I` to the failure state `CC` where Bob's approval field in `requests CC` gets evaluated by the credit company `I` as negative "False". The evaluation steps are:

`I` $\rightarrow$ `C` : Bob puts in a credit request; this is represented by a put action. So, the state `C` has `(''Bob'', None)` $\in$ `requests C`.

`C` $\rightarrow$ `CC` : the credit institute `CI` evaluates the request represented as an `eval` action with the result of the evaluation left in `requests CC`. So, the state `CC` has `(''Bob'', Some(False))` $\in$ `requests CC`.

To derive the final failure state `CC`, the credit institute has applied the `bb` function as `Some((bb C) d)` which evaluates Bob's request as `Some(False)` (see rule `eval` in Section 3.2).

α.2 Next, the PCR algorithm finds an initial precondition that yields the desirable outcome in a closest state using counterfactuals. The closest state is given as `Ca` which differs from `C` in that Bob has a higher salary of 40K as opposed to 35K as in `C`. The state `Ca` is reachable: Bob first applies for a promotion via the action `get`. From the state `Ca`, Bob puts in the credit

[3] It is important to note that we request really only input output pairs and not a mathematical description of the black box. This is in contrast to the stronger assumptions made in the literature, for example [25] (see also the discussion in Section 5.1)

application leading to CCa, before the credit institute CI evaluates leading to CCCa. We see that now with the increased salary of 40K, Bob receives a credit approval.

α.3 The next step of the PCR algorithm is generalisation. We want to investigate whether the salary of 40K is a sufficient precondition in general (for all actors) to explain why the bb algorithm approves the credit. When we try to prove according to Step 3 that this is the case, the attack tree analysis proves the opposite.

$$\texttt{M} \vdash \ \texttt{EF \{ s} \in \texttt{states M.} \ \ \texttt{pc}_i\texttt{(''Alice'',s)} \wedge \neg \ \texttt{DO(''Alice'', s) \}}$$

It turns out that Alice who already has a salary of 40K doesn't get the credit approval. She lives, however, in London's district SE1 unlike Bob who lives in N3. Following thus Step 3.(a) we need to go to another iteration and go back to Step 2. to refine the precondition by counterfactuals.

β.2 In this β-run, we now have the state s where Alice doesn't get the approval. According to Step 2.(a), we find a counterfactual state as the one in which Alice first moves to N3. The new precondition now is created by adding the additional Δ_0 as $\texttt{A} \ @_s \ \texttt{N3}$.

$$\texttt{pc}_{i+1} \ \texttt{(A, s) := salary A s >= 40K} \wedge \texttt{A} \ @_s \ \texttt{N3}$$

β.3 Going to Step 3 again in this β-run, now the proof of the generalisation succeeds.

$$\forall \ \texttt{A. M} \vdash \ \texttt{AG \{s} \in \texttt{states M. pc}_{i+1} \ \texttt{(A, s)} \longrightarrow \ \texttt{EF \{s. DO(A, s)\}\}}$$

4.3 Discussion

With respect to the explanation, the algorithm finishes with the precondition

$$\texttt{pc}_{i+1} \ \texttt{(A, s) := salary A s >= 40K} \wedge \texttt{A} \ @_s \ \texttt{N3}$$

for any A of type actor. Although we terminated the algorithm there, we could have entered another cycle by extending the list of parameters of the precondition adding the location. The generalisation in Step 3 would have triggered the new cycle with providing a precise precondition Δ to specify which locations are sufficient for a desirable outcome. For the sake of conciseness of the exposition, we omit this additional round. But we nevertheless want to discuss it here as it sheds an interesting light onto the evaluation in particular with respect to the GDPR relevance.

It turns out that often there is a bias in the data that has been used to train the black box algorithm. For our case study, we deliberately used such an example to show its potential use for the logical explanation we provide. Since we give a general rule that formally describes and explains the decision process based on actual features of the context of the world. Here, the full run of the PCR algorithm would reveal that for postcodes of London areas in which predominantly black population lives, the salary has to be higher to gain credit

approval. While our example is synthesized, biases like this are known to be implicit in data sets because of the data workers who provided the training data classifications. A very important contribution of our explicit logical model is thus to reveal such biases that are implicit in black box AI algorithms for data evaluation.

How now is this relevant for the GDPR discussion of rights of explanation? The three purposes of explanation that have been identified by Wachter et al [25] (see Section 2.2) with respect to the GDPR are all met by our explanation algorithm.

- *Understand decisions*: the explicit model of context in IIIf contains the rules of the state transition providing the details of each step. The attack trees that produce the traces leading to failure states thus give detailed explanations how the decisions have been arrived at. The algorithm finally produces a general logical rule containing the precondition that explains precisely within the detailed application context what are the relevant facts for decisions.
- *Contest decisions*: the attack trees are explanation trees showing how the decision has been made. The general rule with the precondition provides a means for contesting a decision as it allows to check the decision criteria and reveal potential biases.
- *Alter Future Decisions*: the general rule and its precise precondition allow to read out what are the criteria that can be used as a guideline to alter future decisions. Moreover, the IIIf Kripke model can be explicitly used to simulate the outcome of behaviour by simulating actions of the state transition rules to arrive at favorable states.

5 Related Work and Conclusions

5.1 Related Work

Vigano and Magazzeni [24] argue that explainability is not only needed for AI but as well for security. They use the notion of *XSec* or *Explainable Security* and provide a research agenda for explainability in security centered around the "Six Ws" of XSec: Who? What? Where? When? Why? And hoW? Our point of view is quite similar to Vigano's and Magazzeni's but we emphasize the technical side of explanation using interactive theorem proving and the Isabelle Infrastructure framework, while they focus on differentiating the notion of explanation from different aspects, for example, stake holders, system view, and abstraction levels. Their paper is a position paper that produces a range of research questions illustrating them on examples and showing up potential avenues for future research while we address a very specific way of providing a solution for explanation using automated reasoning with IIIf.

Relevant for the application of the IIIf to the task of explanation is (a) that attack trees resemble explanation trees and (b) that developing a system using attack trees using the RR-cycle resembles the process of generalizing local rules by precondition refinement. Belle and Papantonis [2] already describe how to

infer local explanations from counterfactuals using quantitative information. A local explanation corresponds to a rule. "Robustness" of the rule means that similar instances will get the same outcome – a starting point for developing more general rules. Their approach strongly inspired our development of the PCR cycle because – in analogy to using attack trees as a trigger for the RR-cycle – counterfactuals are now used to guide the development of general "robust" rules. Nevertheless our work provides a precise process of precondition refinement within Isabelle as well as a framework that extends the IIIf to support explanation within rich human centric models.

Wachter et al [25] is a paper that strongly inspired our work. We used their analysis of explanation and the GDPR as requirements for our analysis and validation of the PCR cycle. However, there are a number of differences. Wachter advocates strongly the use of counterfactuals as fully sufficient for the explanation of black box decision procedures. Nevertheless, they use a function f_w [25, p. 854] as explicit input to their optimiser that allows the computation of counterfactuals at any given data points. Thus it is not really a black box algorithm they assume. Consequently, in their demonstration example [25, Appendix A], they easily outperform a very simple explanation method like LIME that uses linear regression. Another difference to our work is that they only consider quantitative functions, like salary. Context features, like location, ethnicity, etc, that are central to our logical method of building a complete rule explanation are not represented.

5.2 Conclusions

In this paper, we have shown how the RR-cycle of the IIIf can be adapted to provide a method for iteratively extracting an explanation by interleaving attack tree analysis with precondition refinement. This precondition refinement (PCR) cycle finally yields a general rule that describes the decision taken by a black box algorithm produced by AI. Since it is a logical rule within a rich context of an infrastructure model of the application scenario, it provides transparency, We argue that the three purposes of the right of explanation of the GDPR of understanding, contesting and altering a decision given by an automated AI decision procedure are supported by the PCR cycle.

The PCR cycle only needs to slightly adapt the RR-cycle by implementing an algorithm to define a methodology for interleaving attack tree analysis with a step by step refinement of a precondition using counterfactuals. Responsible for the ease of this adaptation is the first class representation of attack trees in the IIIf. That is, the existing Correctness and Completeness result of attack trees with respect to the CTL logic defined over Kripke structures allows changing between attack trees and CTL EF formulas. Thus attack trees can be reused as explanation trees because they explain how failure states are reached. This in turn allows the construction of counterfactuals that guide the refinement of the precondition. This paper has validated the algorithm of the PCR cycle by a case study of credit scoring. Further work should address to what extent finding the counterfactuals and thereby the refined precondition can be automated.

References

1. A. B. Arrieta, N. Díaz-Rodríguez, A. B. Javier Del Ser, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molinaa, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, (58):82–115, 2020.
2. V. Belle and I. Papantonis. Principles and practice of explainable machine learning. *CoRR*, abs/2009.11698, 2020.
3. S. Bull. London boroughs mapped and ranked by residents' credit scores - how money-savvy is your area? Accessed 22.7.2022.
4. CHIST-ERA. Success: Secure accessibility for the internet of things, 2016. http://www.chistera.eu/projects/success.
5. L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. A. Specter, and L. Kagal. Explaining explanations: An approach to evaluating interpretability of machine learning. *CoRR*, abs/1806.00069, 2018.
6. T. is Money. How well do your neighbours manage their money? interactive map reveals average credit scores by postcode. Accessed 22.7.2022, `https://www.thisismoney.co.uk/money/cardsloans/article-3273996/How-neighbours-manage-money-Interactive-map-reveals-average-credit-scores-postcode.html`.
7. F. Kammüller. A proof calculus for attack trees. In *Data Privacy Management, DPM'17, 12th Int. Workshop*, volume 10436 of *LNCS*. Springer, 2017. Co-located with ESORICS'17.
8. F. Kammüller. Attack trees in isabelle. In *20th International Conference on Information and Communications Security, ICICS2018*, volume 11149 of *LNCS*. Springer, 2018.
9. F. Kammüller. Formal modeling and analysis of data protection for gdpr compliance of iot healthcare systems. In *IEEE Systems, Man and Cybernetics, SMC2018*. IEEE, 2018.
10. F. Kammüller. Attack trees in isabelle extended with probabilities for quantum cryptography. *Computer & Security*, 87, 2019.
11. F. Kammüller. Combining secure system design with risk assessment for iot healthcare systems. In *Workshop on Security, Privacy, and Trust in the IoT, SPTIoT'19, colocated with IEEE PerCom*. IEEE, 2019.
12. F. Kammüller. Dependability engineering in isabelle, 2021. arxiv preprint, `http://arxiv.org/abs/2112.04374`.
13. F. Kammüller. Explanation by automated reasoning using the isabelle infrastructure framework, 2021. arxiv preprint, `http://arxiv.org/abs/2112.14809`.
14. F. Kammüller. Isabelle Insider and Infrastructure framework with Kripke strutures, CTL, attack trees, security refinement, and examples including IoT, GDPR, QKD, social networks, and credit scoring, 2022. Available at `https://github.com/flokam/IsabelleAT`.
15. F. Kammüller and C. M. Alvarado. Exploring rationality of self awareness in social networking for logical modeling of unintentional insiders, 2021. arxiv preprint, `http://arxiv.org/abs/2111.15425`.
16. F. Kammüller and M. Kerber. Applying the isabelle insider framework to airplane security. *Science of Computer Programming*, 206, 2021.
17. F. Kammüller, M. Kerber, and C. Probst. Insider threats for auctions: Formal modeling, proof, and certified code. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 8(1), 2017.

18. F. Kammüller and B. Lutz. Modeling and analyzing the corona-virus warning app with the isabelle infrastructure framework. In *20th International Workshop of Data Privacy Management, DPM'20*, volume 12484 of *LNCS*. Springer, 2020. Co-located with ESORICS'20.
19. F. Kammüller and U. Nestmann. Inter-blockchain protocols with the isabelle infrastructure framework. In *Formal Methods for Blockchain, 2nd Int. Workshop, colocated with CAV'20*, Open Access series in Informatics. Dagstuhl publishing, 2020. To appear.
20. F. Kammüller, M. Wenzel, and L. C. Paulson. Locales – a sectioning concept for Isabelle. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Thery, editors, *Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs'99*, volume 1690 of *LNCS*. Springer, 1999.
21. A. C. Myers and B. Liskov. Complete, safe information flow with decentralized labels. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 1999.
22. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer-Verlag, 2002.
23. W. Pieters. Explanation and trust: What to tell the user in security and ai? *Ethics and Information Technology*, 13(1):53–64, 2011.
24. L. Viganó and D. Magazzeni. Explainable security. In *IEEE European Symposium on Security and Privacy Workshops, EuroS&PW*. IEEE, 2020.
25. S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanantions without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2), 2018.