Backward Responsibility in Transition Systems Using General Power Indices

Christel Baier[1,2], Roxane van den Bossche[3], Sascha Klüppelholz[1], Johannes Lehmann[1,2], Jakob Piribauer[1] *

[1]TU Dresden, Germany
[2]Centre for Tactile Internet with Human-in-the-Loop (CeTI)
[3]Université Paris-Saclay, ENS Paris-Saclay, France
{christel.baier, sascha.klueppelholz, johannes_alexander.lehmann, jakob.piribauer}@tu-dresden.de,
roxane.van_den_bossche@ens-paris-saclay.fr

Abstract

To improve reliability and the understanding of AI systems, there is increasing interest in the use of formal methods, e.g. model checking. Model checking tools produce a counterexample when a model does not satisfy a property. Understanding these counterexamples is critical for efficient debugging, as it allows the developer to focus on the parts of the program that caused the issue.

To this end, we present a new technique that ascribes a responsibility value to each state in a transition system that does not satisfy a given safety property. The value is higher if the non-deterministic choices in a state have more power to change the outcome, given the behaviour observed in the counterexample. For this, we employ a concept from cooperative game theory – namely general power indices, such as the Shapley value – to compute the responsibility of the states.

We present an optimistic and pessimistic version of responsibility that differ in how they treat the states that do not lie on the counterexample. We give a characterisation of optimistic responsibility that leads to an efficient algorithm for it and show computational hardness of the pessimistic version. We also present a tool to compute responsibility and show how a stochastic algorithm can be used to approximate responsibility in larger models. These methods can be deployed in the design phase, at runtime and at inspection time to gain insights on causal relations within the behavior of AI systems.

1 Introduction

Due to the ever-growing demand for reliable, trustworthy AI systems, research on ways to incorporate formal methods for the verification of such systems is becoming more and more important (see e.g. Seshia, Sadigh, and Sastry (2022)). AI systems are in general instances of parallel systems that operate in complex environments. As such, techniques such as model checking are suitable for this setting. The aim of model checking is to analyse a given system to prove automatically that this system satisfies some property. For example, we might require certain states to never be reached, because they correspond to an undesired event. When a model does not satisfy a safety property, the model checker returns a *counterexample*, which is an execution of the model that violates the property (see Baier and Katoen (2008) for more details). While this demonstrates that an error exists, it is not obvious which part of the model is *responsible* for the error. Some states in the counterexample may be unable to change the outcome, whereas others can ensure that the safety property is satisfied.

We distinguish between *forward* and *backward* responsibility (Van de Poel 2011). In the forward case, responsibility depends only on the model, whereas in the backward case, it also takes a specific counterexample into account. Existing causality-based approaches to backward responsibility include the use of distance metrics (Groce et al. 2006), mutation-based techniques (Beer et al. 2012), event-order logic (Leitner-Fischer and Leue 2013) and hyperproperties (Coenen et al. 2022).

In this paper, we introduce an intuitive, game-based approach to compute *backward* responsibility. Our approach is inspired by Mascle et al. (2021) where the Shapley value (Shapley et al. 1953) is used to determine a numeric value of *forward* responsibility. For a more recent discussion of the Shapley value, we refer the reader to Laruelle and Valenciano (2001). The Shapley value is also used for responsibility attribution by Baier, Funke, and Majumdar (2021), who present a technique that operates on game trees.

Incorporating a counterexample allows us to analyse a specific fault in the system. Moreover, we extend our definitions to general power indices, also known as semivalues (Dubey, Neyman, and Weber 1981), of which the Shapley value is an instance.

To illustrate our scenario, consider the railway network depicted in Figure 1 with three switches s_1, s_2 and s_3. The goal is to route the train to the destination ✓. However, it can also be routed to the unfinished line ✗, which causes an accident. We investigate how the switches share the responsibility for the outcome.

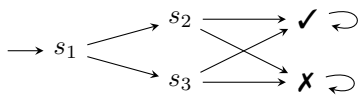


Figure 1: Railway network with three switches s_1, s_2 and s_3. If a train is routed to ✗, an accident occurs.

*Authors are listed in alphabetical order

In the forward responsibility, s_2 and s_3 would have the same responsibility due to the symmetry of the system. Now consider a scenario where an accident has occurred after the train took the path $s_1 s_2$✗. It is now desirable to ascribe more responsibility to s_2 than to s_3.

To compute the exact responsibility, we determine which coalitions of states can ensure that ✗ is not reached. For example, given that the train travelled from s_1 to s_2, $\{s_2\}$ suffices: if it routes the train to ✓ instead of ✗, the accident is averted. For $\{s_1\}$, the answer is less clear. It can route the train to s_3 instead, but we have not observed the behaviour of s_3 in the system. We therefore distinguish between two types of responsibility: For *optimistic responsibility*, we assume that these states help us in trying to satisfy the safety property. For *pessimistic responsibility*, on the other hand, we assume that they attempt to violate the property. For optimistic responsibility, $\{s_1\}$ can avoid ✗, whereas for pessimistic responsibility, it cannot (but $\{s_1, s_3\}$ can). We analyse both types of backward responsibility in this article.

Main Contributions: The main contributions of this paper are: We give a definition of backward responsibility based on Mascle et al. and define the novel notion of optimistic responsibility in Section 3. We analyse the complexity of pessimistic and optimistic responsibility in Section 4. We provide an implementation, show how a stochastic algorithm can be used to analyse large models and evaluate our implementation with several experiments in Section 5. A full version of this paper including detailed proofs is available at Baier et al. (2024).

Other Related Work: There are several other approaches to assess backward responsibility. In addition to a counterexample, many of these require one or more passing runs, which are then compared. An example of this is *Delta debugging* (Zeller 2002). Ball, Naik, and Rajamani (2003) take an intraprocedural approach and also use the information to identify multiple errors that may be present in a single model. Groce et al. (2006) use distance metrics to identify the differences between passing runs and a failing run and present them to the user as an explanation. A similar approach using nearest-neighbor queries is presented by Renieres and Reiss (2003).

On the other hand, the technique of Wang et al. (2006) does not require additional runs and instead analyses the counterexample using weakest-precondition reasoning.

Similar to our technique and to Mascle et al. (2021), Baier, Funke, and Majumdar (2021) use the Shapley value and game theory to analyse responsibility. They operate on games in tree form and also distinguish between forward and backward responsibility.

A related concept is *blameworthiness* as defined by Halpern and Kleiman-Weiner (2018) is a notion aiming to define *moral* responsibility of agents. Our notion of responsibility does not include any moral considerations. Nevertheless, notions of blameworthiness can be defined using the technical vehicle of power indices as used for our notion of responsibility, such as the Shapley value used in Friedenberg and Halpern (2019).

2 Preliminaries

Transition systems. A *transition system* is a tuple $TS = (S, \to, s_0, \lightning)$ where S is a finite set of *states*, $\to$ is the *transition relation* on S, $s_0 \in S$ is the *initial state* and $\lightning \subseteq S$ is the set of *bad states*. A *run* on TS is an infinite sequence of states $\rho = \rho_0 \rho_1 \ldots \in S^\omega$, where $\rho_0 = s_0$ and $\forall i \in \mathbb{N}$ $\rho_i \to \rho_{i+1}$.

We call $\rho = \rho_0 \ldots \rho_k \in S^*$ a counterexample if it is the prefix of a run, $\rho_k \in \lightning$, $\rho_i \notin \lightning$ for all $i \in \{0, \ldots, k-1\}$ and $\rho_i \neq \rho_j$ for all $i \neq j$, i.e. they are loop-free.

Cooperative games. Let X be a finite set of *players*. A *cooperative game* on X is a function $v\colon 2^X \to \mathbb{R}$ that associates a value to each subset of X. We call $C \subseteq X$ a *coalition* and $v(C)$ the *gain* of the coalition C. The set of games on X is denoted by G^X.

We call v *simple* if $v(C) \in \{0, 1\}$ for all $C \subseteq 2^X$. A coalition $C \subseteq 2^X$ is *winning* if $v(C) = 1$. (C, s) forms a *critical pair* if $C \cup \{s\}$ is winning and C is not.

General power indices. We call $p = (p_0, \ldots, p_{n-1})$ a *weight vector* if $\sum_{k=0}^{n-1} \binom{n-1}{k} p_k = 1$.

Let X be a finite set of players with $n := |X|$. Then $\mathcal{R}\colon G^X \to X \to \mathbb{R}$ is a *general power index* if there exists a weight vector $p = (p_0, \ldots, p_{n-1})$ such that, for any game $v \in G^X$ and player $i \in X$, we have

$$\mathcal{R}(v, i) = \sum_{C \subset X \setminus \{i\}} p_{|C|}[v(C \cup \{i\}) - v(C)]$$

where we write $\mathcal{R}(v, i)$ instead of $(\mathcal{R}(v))(i)$.

We note that for every general power index $\mathcal{R}$, only one such weight vector $p = (p_0, \ldots, p_{n-1})$ exists and define $\texttt{Weights}_i(\mathcal{R}) = p_i$.

This definition corresponds to the characterisation of semivalues for games on finite sets of players given by Dubey, Neyman, and Weber (1981). The linear games they use are isomorphic to functions from X to $\mathbb{R}$, so our definition is equivalent to their definition and characterisation.

Shapley value, Banzhaf value. Let X be a finite set of players and let v be a game on S. The *Shapley value* is the general power index $\mathcal{S}$ with $\texttt{Weights}_i(\mathcal{S}) = \frac{(n-i-1)!i!}{n!}$. The *Banzhaf value* is the general power index $\mathcal{B}$ with $\texttt{Weights}_i(\mathcal{B}) = \frac{1}{2^{n-1}}$.

We use the following well-known characterisation of the Shapley value in several proofs.

Lemma 1. Let X be a finite set of players. We have $\mathcal{S}(v, x) = \sum_{\pi \in \Pi_X} (v(\pi_{\geq x}) - v(\pi_{>x}))$ for any $v \in G^X$ and $x \in X$, where Π_X is the set of permutations of X and $\pi_{\triangleright x} := \{y \in X \mid \pi(y) \triangleright \pi(x)\}$ for $\triangleright \in \{\geq, >\}$.

Games. A *game arena* is a tuple $(S_{\textit{Safe}}, S_{\textit{Reach}}, \to, s_0)$ where $S_{\textit{Safe}}$ is the set of states controlled by *Safe*, $S_{\textit{Reach}}$ is the set of states controlled by *Reach* (and we write $S := S_{\textit{Safe}} \dot\cup S_{\textit{Reach}}$), $\to$ is the transition relation on S and $s_0 \in S$ is the initial state.

A *game* consists of a game arena $(S_{\textit{Safe}}, S_{\textit{Reach}}, \to, s_0)$ and a *winning condition* $\Omega \subseteq S^\omega$. A *play* $\rho \in S^\omega$ is an infinite sequence $\rho_0 \rho_1 \ldots$ such that $\rho_0 = s_0$ and $(\rho_i, \rho_{i+1}) \in \to$ for

all $i \in \mathbb{N}$. A play ρ is *winning* for *Safe* if $\rho \in \Omega$, otherwise it is winning for *Reach*. A strategy for *Safe* is a function $\sigma\colon S_{\mathit{Safe}} \to S$ with $(s, \sigma(s)) \in \to$ for all $s \in S_{\mathit{Safe}}$ (and strategies for *Reach* are defined similarly). A pair of strategies $(\sigma_{\mathit{Safe}}, \sigma_{\mathit{Reach}})$ for *Safe* and *Reach* induces a play $\rho = \rho_0\rho_1\ldots$ with $\rho_{i+1} = \sigma_{\mathit{Safe}}(\rho_i)$ if $\rho_i \in S_{\mathit{Safe}}$ and $\rho_{i+1} = \sigma_{\mathit{Reach}}(\rho_i)$ otherwise. A strategy for *Safe* is winning if, for all strategies of *Reach*, the induced play is winning for *Safe* (and winning strategies for *Reach* are defined similarly).

Given $\lightning \subseteq S$, a safety winning condition has the form $\Omega_{\lightning} = \{\rho \mid \forall i \in \mathbb{N}\colon \rho_i \notin \lightning\}$. A safety game consists of a game arena and $\lightning \subseteq S$ and we write $(S_{\mathit{Safe}}, S_{\mathit{Reach}}, \to, s_0, \lightning)$ instead of $((S_{\mathit{Safe}}, S_{\mathit{Reach}}, \to, s_0), \Omega_{\lightning})$.

3 Optimistic and Pessimistic Responsibility

To use general power indices, we must formulate the responsibility problem as a cooperative game. Therefore, we need to define a value function which takes a coalition of states C and returns 0 or 1 depending on whether C can avoid $\lightning$.

First, we define a safety game where the run of the counterexample is *engraved*: For every state that is on the run and not in C, we remove the outgoing transitions except for the transition that follows the run.

Definition 2 (From transition systems to safety games)**.** Let $TS = (S, \to, s_0, \lightning)$ be a transition system, $\rho = \rho_0 \ldots \rho_k$ a counterexample and $C \subseteq S$. We define $\mathcal{G}_\rho^{TS}(C) = (C, S \setminus C, \to', s_0, \lightning)$, where $\rho_i \to' \rho_{i+1}$ for $i \in \{0, \ldots, k-1\}$ if $\rho_i \notin C$ and $s \to' s'$ if $s \to s'$ and if $s \notin \rho$ or $s \in C$.

We now use this to define a cooperative game.

Definition 3 (Optimistic and pessimistic cooperative games)**.** Let $TS = (S, \to, s_0, \lightning)$ be a transition system, ρ a counterexample and $C \subseteq S$. Then the optimistic cooperative game $v_{\mathtt{opt}}^{TS,\rho}$ is defined as

$$v_{\mathtt{opt}}^{TS,\rho}(C) = \begin{cases} 1 & \text{if player } \textit{Safe} \text{ wins } \mathcal{G}_\rho^{TS}(C \cup (S \setminus \rho)) \\ 0 & \text{otherwise} \end{cases}$$

and the pessimistic cooperative game $v_{\mathtt{pes}}^{TS,\rho}$ is defined as

$$v_{\mathtt{pes}}^{TS,\rho}(C) = \begin{cases} 1 & \text{if player } \textit{Safe} \text{ wins } \mathcal{G}_\rho^{TS}(C) \\ 0 & \text{otherwise.} \end{cases}$$

In the following, let $TS = (S, \to, s_0, \lightning)$ be a transition system and $\rho = \rho_0 \ldots \rho_k$ a counterexample. We set $n \coloneqq |S|$. We write $v_{\mathtt{opt}}$ instead of $v_{\mathtt{opt}}^{TS,\rho}$ and $v_{\mathtt{pes}}$ instead of $v_{\mathtt{pes}}^{TS,\rho}$.

Example 4. Consider the railway network from Figure 1 with $\lightning = \{\text{✗}\}$ and $\rho = s_1 s_2 \text{✗}$. Let $C = \varnothing$ (and thus $C \cup (S \setminus \rho) = \{s_3, \text{✓}\}$). In both $\mathcal{G}_\rho^{TS}(C)$ and $\mathcal{G}_\rho^{TS}(C \cup (S \setminus \rho))$, the only transition from s_1 goes to s_2 and the only transition from s_2 goes to ✗. Therefore, *Safe* cannot win and thus $v_{\mathtt{opt}}(C) = v_{\mathtt{pes}}(C) = 0$.

Now let $C' = \{s_1\}$ and therefore $C' \cup (S \setminus \rho) = \{s_1, s_3, \text{✓}\}$. Then *Safe* wins $\mathcal{G}_\rho^{TS}(C' \cup (S \setminus \rho))$ by playing $s_1 s_3 \text{✓}^\omega$, which are all states controlled by her. However, she loses $\mathcal{G}_\rho^{TS}(C')$, because either s_2 or s_3 is reached. *Reach* can then move to ✗. Therefore, $v_{\mathtt{opt}}(C') = 1$ and $v_{\mathtt{pes}}(C') = 0$.

Proposition 5. Let $C \subseteq S$ and $s \in S$. Then we have $v_{\mathtt{opt}}(C) \geq v_{\mathtt{pes}}(C)$.

Proof. If $v_{\mathtt{pes}}(C) = 1$, then *Safe* wins $\mathcal{G}_\rho^{TS}(C)$. Because reachability games are monotonic, *Safe* also wins $\mathcal{G}_\rho^{TS}(C \cup (S \setminus \rho))$ and thus $v_{\mathtt{opt}}(C) = 1$. If $v_{\mathtt{pes}}(C) = 0$, then the inequality holds for every value of $v_{\mathtt{opt}}(C)$. □

The forward view, as presented by Mascle et al. (2021), is closely related to our pessimistic definition. This is because the optimistic definition does not work without a counterexample. Every state in C would belong to *Safe*, but every state not in C would also be given to *Safe* under the optimistic assumption, so the outcome of the game would not be affected by the value of C.

Definition 6 (Responsibility)**.** Let $\mathcal{R}$ be a general power index on G^S. The *optimistic responsibility of state s with respect to $\mathcal{R}$* is $\mathcal{R}(v_{\mathtt{opt}}, s)$ and the *pessimistic responsibility of state s with respect to $\mathcal{R}$* is $\mathcal{R}(v_{\mathtt{pes}}, s)$.

For example, the pessimistic responsibility of s with respect to the Banzhaf value is $\mathcal{B}(v_{\mathtt{pes}}, s)$.

Example 7. We omit ✓ and ✗ from the coalitions, as control over them never affects the outcome. In the pessimistic case, (C, s_2) forms a critical pair for $C \in \{\varnothing, \{s_1\}, \{s_3\}\}$. Therefore, the Shapley responsibility of s_2 is $\mathcal{S}(v_{\mathtt{pes}}, s_2) = 1 \cdot \mathtt{Weights}_0(\mathcal{S}) + 2 \cdot \mathtt{Weights}_1(\mathcal{S}) = \frac{2}{3}$. For s_1, only $(\{s_3\}, s_1)$ forms a critical pair and for s_3, only $(\{s_1\}, s_3)$ forms a critical pair, so we have $\mathcal{S}(v_{\mathtt{pes}}, s_1) = \mathcal{S}(v_{\mathtt{pes}}, s_3) = 1 \cdot \mathtt{Weights}_1(\mathcal{S}) = \frac{1}{6}$. The full responsibility values for the optimistic and pessimistic case are given in Table 1. While it may seem counterintuitive that s_3 has positive responsibility even though it was not involved in reaching ✗, this makes sense upon closer inspection: s_1 can only avoid ✗ if s_3 helps, so it is natural that they share the responsibility.

State s	s_1	s_2	s_3	✓	✗
$\mathcal{S}(v_{\mathtt{pes}}, s)$	$1/6$	$2/3$	$1/6$	0	0
$\mathcal{S}(v_{\mathtt{opt}}, s)$	$1/2$	$1/2$	0	0	0

Table 1: Optimistic and pessimistic Shapley responsibilities for the train example from Figure 1.

Note that the sum of the responsibility of s_1, s_2 and s_3 is 1. This is a general property of the Shapley value (but not of other general power indices, such as the Banzhaf value).

Proposition 8. If there exists a path in TS which does not reach $\lightning$, then we have

$$\sum_{s \in S} \mathcal{S}(v_{\mathtt{opt}}, s) = \sum_{s \in S} \mathcal{S}(v_{\mathtt{pes}}, s) = 1.$$

Proof. Let $v \in \{v_{\mathtt{opt}}, v_{\mathtt{pes}}\}$. By Lemma 1, we have $\mathcal{S}(v,s) = \sum_{\pi \in \Pi_S} (v(\pi_{\geq s}) - v(\pi_{>s}))$. Because ρ is a counterexample, we have $v(\varnothing) = 0$ and because there exists a path that does not reach $\lightning$, we have $v(S) = 1$. As v is monotonic, for each permutation π, there is exactly one state s with $v(\pi_{\geq s}) - v(\pi_{>s}) = 1$. We call this state $\mathtt{Swi}(\pi)$. For all other states, the difference is 0. Therefore, we have

$$\begin{aligned}\sum_{s\in S} \mathcal{S}(v,s) &= \sum_{s\in S} \frac{1}{n!} \sum_{\pi\in\Pi_S} (v(\pi_{\geq s}) - v(\pi_{>s})) \\ &= \frac{1}{n!} \sum_{\pi\in\Pi_S} \sum_{s\in S} (v(\pi_{\geq s}) - v(\pi_{>s})) \\ &= \frac{1}{n!} \sum_{\pi\in\Pi_S} \left(v(\pi_{\geq \mathtt{Swi}(\pi)}) - v(\pi_{>\mathtt{Swi}(\pi)})\right) \\ &= \frac{1}{n!} \sum_{\pi\in\Pi_S} 1 = 1. \end{aligned}$$ □

States that only have a single outgoing transition always have responsibility 0. On the other hand, there can be states with multiple outgoing transitions that have responsibility 0, e.g. if all outgoing paths from a state eventually reach $\lightning$.

4 Algorithms

We investigate the complexity of two decision problems and one counting problem. Each problem has an optimistic variant (where $v^{TS,\rho} = v^{TS,\rho}_{\mathtt{opt}}$) and a pessimistic variant (where $v^{TS,\rho} = v^{TS,\rho}_{\mathtt{pes}}$). A general power index $\mathcal{R}$ is encoded by encoding $\mathtt{Weights}_i(\mathcal{R})$ for $i \in \{0,\dots,n-1\}$.

Positivity Problem

$$\begin{cases}\text{Input:} & TS = (S, \to, s_0, \lightning),\ \text{counterexample } \rho, \\ & s \in S, \text{general power index } \mathcal{R} \\ \text{Output:} & \text{Is } \mathcal{R}(v^{TS,\rho}, s) > 0?\end{cases}$$

Threshold Problem

$$\begin{cases}\text{Input:} & TS = (S, \to, s_0, \lightning),\ \text{counterexample } \rho, \\ & s \in S, \text{general power index } \mathcal{R}, t \in [0,1) \\ \text{Output:} & \text{Is } \mathcal{R}(v^{TS,\rho}, s) > t?\end{cases}$$

Computation Problem

$$\begin{cases}\text{Input:} & TS = (S, \to, s_0, \lightning),\ \text{counterexample } \rho, \\ & s \in S, \text{general power index } \mathcal{R} \\ \text{Output:} & \text{What is the value of } \mathcal{R}(v^{TS,\rho}, s)?\end{cases}$$

Optimistic Responsibility

We show that the optimistic responsibility is characterised by a simple function. This characterisation then yields an efficient algorithm for computing optimistic responsibility.

For the characterisation, we first define the sets of winning and responsible states.

Definition 9 ($\mathtt{WS}_{\mathtt{opt}}, \mathtt{RS}_{\mathtt{opt}}(\mathcal{R})$)**.** Let $\mathcal{R}$ be a general power index. The set of *winning states* is $\mathtt{WS}_{\mathtt{opt}} = \{s \in S \mid v_{\mathtt{opt}}(\{s\}) = 1\}$, i.e. the set of the states that can win on their own. The set of *responsible states* is $\mathtt{RS}_{\mathtt{opt}}(\mathcal{R}) = \{s \in S \mid \mathcal{R}(v_{\mathtt{opt}}, s) > 0\}$, i.e. the set of the states that have a strictly positive responsibility.

The following lemma relates states that can win on their own and states with positive responsibility.

Lemma 10. Let $\mathcal{R}$ be a general power index. For any $s \in S$, we have $\mathcal{R}(v_{\mathtt{opt}}, s) > 0 \implies v_{\mathtt{opt}}(\{s\}) = 1$, i.e. $\mathtt{RS}_{\mathtt{opt}}(\mathcal{R}) \subseteq \mathtt{WS}_{\mathtt{opt}}$.

If $\mathtt{Weights}_0(\mathcal{R}) > 0$, the converse implication $\mathcal{R}(v_{\mathtt{opt}}, s) > 0 \impliedby v_{\mathtt{opt}}(\{s\}) = 1$ also holds and thus $\mathtt{RS}_{\mathtt{opt}}(\mathcal{R}) = \mathtt{WS}_{\mathtt{opt}}$.

Corollary 11. If $\mathtt{Weights}_i(\mathcal{R}) > 0$ for $i \in \{0,\dots,n-1\}$, then $\mathcal{R}(v_{\mathtt{opt}}, s) > 0 \implies \mathcal{R}(v_{\mathtt{pes}}, s) > 0$.

Using Lemma 10, the following theorem then shows that responsibility in the optimistic case is a yes-or-no question.

Theorem 12 (Characterisation of optimistic responsibility)**.** Let $\mathcal{R}$ be a general power index, then we have

$$\mathcal{R}(v_{\mathtt{opt}}, s) = \begin{cases} K & \text{if } s \in \mathtt{WS}_{\mathtt{opt}} \\ 0 & \text{otherwise}\end{cases}$$

with $K = \sum_{i=0}^{n-w} \binom{n-w}{i} \cdot \mathtt{Weights}_i(\mathcal{R})$ and $w \coloneqq |\mathtt{WS}_{\mathtt{opt}}|$. Additionally, we have $\mathtt{WS}_{\mathtt{opt}} \subseteq \rho$.

The relevant question in the optimistic case is therefore "Is s in $\mathtt{WS}_{\mathtt{opt}}$?", i.e. "Can s win on its own?".

Proof. Let $s \in \mathtt{WS}_{\mathtt{opt}}$ and let $p = (p_0, \dots, p_{n-1})$ with $p_i = \mathtt{Weights}_i(\mathcal{R})$. Then

$$\mathcal{R}(v_{\mathtt{opt}}, s) = \sum_{C \subseteq S\setminus\{s\}} p_{|C|}[v_{\mathtt{opt}}(C \cup \{s\}) - v_{\mathtt{opt}}(C)].$$

Let $C \subseteq S \setminus \{s\}$. Lemma 10 implies that $v_{\mathtt{opt}}(C \cup \{s\}) - v_{\mathtt{opt}}(C) = 1$ if and only if $C \cap \mathtt{WS}_{\mathtt{opt}} = \varnothing$ and $v_{\mathtt{opt}}(C \cup \{s\}) - v_{\mathtt{opt}}(C) = 0$ otherwise. In the former case, $C \subseteq S \setminus \mathtt{WS}_{\mathtt{opt}}$ and $\left|S \setminus \mathtt{WS}_{\mathtt{opt}}\right| = n - w$. Therefore, there are $\binom{n-w}{i}$ coalitions of size i that fulfil the condition and we have

$$\begin{aligned}\mathcal{R}(v_{\mathtt{opt}}, s) &= \sum_{C \subseteq S\setminus \mathtt{WS}_{\mathtt{opt}}} p_{|C|}[v_{\mathtt{opt}}(C \cup \{s\}) - v_{\mathtt{opt}}(C)] \\ &= \sum_{C \subseteq S\setminus \mathtt{WS}_{\mathtt{opt}}} p_{|C|} = \sum_{i=0}^{n-w} \binom{n-w}{i} p_i.\end{aligned}$$

We now show that $\mathtt{WS}_{\mathtt{opt}} \subseteq \rho$. By Lemma 10, if $\mathtt{Weights}_0(\mathcal{R}) > 0$ and $\mathcal{R}(v_{\mathtt{opt}}, s) > 0$, then s can avoid reaching $\lightning$ on its own. If $s \notin \rho$, this is impossible. □

In the case of the Shapley and Banzhaf values, we can give a closed form for the constant K:

Proposition 13. For any $s \in \mathtt{WS}_{\mathtt{opt}}$, we have

$$\mathcal{S}(v_{\texttt{opt}}, s) = \frac{1}{|\texttt{WS}_{\texttt{opt}}|} \quad \text{and} \quad \mathcal{B}(v_{\texttt{opt}}, s) = \frac{1}{2^{|\texttt{WS}_{\texttt{opt}}|-1}}.$$

Table 1 shows that pessimistic responsibility does not satisfy this property, as s_1 and s_2 have different positive responsibilities.

This characterisation now yields an efficient algorithm for optimistic responsibility. We compute the set of winning states $\texttt{WS}_{\texttt{opt}}$ as follows. For each state $s \in S$, we determine whether $\{s\}$ is a winning coalition by constructing $\mathcal{G}_\rho^{TS}(\{s\})$ and solving it with the attractor algorithm (Grädel, Thomas, and Wilke 2002), which has linear runtime. We invoke it $|S|$ times, yielding quadratic runtime. Due to Lemma 10, if $s \notin \texttt{WS}_{\texttt{opt}}$, the responsibility of s is 0. Otherwise, we can compute the responsibility constant K from Theorem 12 and return it. This yields the following theorem.

Theorem 14. The optimistic positivity, threshold and computation problems are solvable in polynomial time.

Pessimistic Responsibility

Computing pessimistic responsibility is more difficult:

Proposition 15. The pessimistic positivity problem is NP-complete.

Proof sketch. To show inclusion in NP, we nondeterministically guess a coalition C and then verify that $\texttt{Weights}_{|C|}(\mathcal{R})(v_{\texttt{pes}}(C \cup \{s\}) - v_{\texttt{pes}}(C)) > 0$. This requires polynomial time and thus, the problem is in NP.

To show NP-hardness, we give a reduction from the forward responsibility positivity problem, which Mascle et al. (2021) have shown to be NP-hard. For this, we take the given transition system TS and and construct a new transition system TS' with new initial state s_0' and edges from s_0' to s_0 and $\lightning$. We choose $\rho = s_0'\bar{s}$ for some $\bar{s} \in \lightning$. Then a coalition C is winning in TS if and only if $C \cup \{s_0'\}$ is winning in TS' and thus, a state has positive responsibility in TS if and only if it has positive responsibility in TS'. □

Proposition 16. The pessimistic threshold problem is NP-hard and in PSPACE.

Proof sketch. NP-hardness follows from the NP-hardness of the positivity problem shown in Proposition 15 by choosing threshold $t = 0$.

To show inclusion in PSPACE, we construct an algorithm that computes the responsibility in polynomial space. For this, we count the number of coalitions of each size that are significant, multiply the counts by the corresponding weights, and then output the sum. □

The pessimistic threshold problem thus lies between NP and PSPACE, but the precise complexity is still open.

Proposition 17. The pessimistic computation problem is #P-hard and can be solved in polynomial space.

Proof sketch. A polynomial-space algorithm is given in Proposition 16. #P-hardness can be shown by reduction from the forward computation problem (Mascle et al. 2021), which is #P-complete. We perform the same construction as in Proposition 15 and show that there is a bijection between winning coalitions in the forward and in the backward view. □

5 Implementation

We have developed a tool that computes optimistic and pessimistic backward responsibility[1] (Lehmann 2024). Our tool implements three techniques for this: It supports exact computation of responsibility, which works well on small models, but has exponential runtime. It can also use a stochastic algorithm that produces a good approximation of responsibility for larger models with thousands of states. Finally, it is possible to group states, which makes results less cluttered and improves runtime significantly.

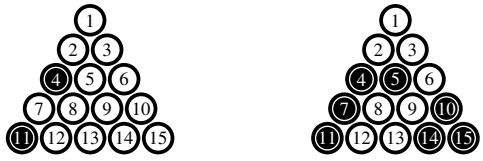
All benchmarks were run on a MacBook Pro running macOS 13.3 with an 8-core M2 chip and 24 GB of memory. The experiments were conducted using the Shapley value, as it is the most widely-used general power index.

Exact Algorithm

To compute responsibility exactly, our tool uses the model checker PRISM (Kwiatkowska, Norman, and Parker 2011) to build the model and check the safety property. PRISM produces a transition system and counterexample as output. Alternatively, this raw model and counterexample can be provided directly by the user. For every coalition and state, we then check whether they form a critical pair. If they do, we increment the result by the value given by the general power index.

We found that, in practice, it is faster to first compute the minimal winning coalitions of the game and then analyse the coalitions. This way, each coalition is only solved once. Both steps can be parallelised with minimal overhead.

Example: Peg Solitaire In the single-player board game *Peg Solitaire* (Bell 2007), the player is presented with a grid of holes (in our case, the grid is triangle-shaped). All but one of the holes are filled with a peg. In each move, the player may pick any peg and jump over an adjacent peg-filled hole into an empty hole behind that. After that, the peg they jumped over is removed. The goal is to remove all but one of the pegs. It is possible to lose by reaching a configuration with multiple pegs where none of the pegs can jump. Such a configuration is depicted in Figure 2a.



(a) Final, losing configuration of the game

(b) Last configuration with positive responsibility

Figure 2: Analysis of a Peg Solitaire game.

[1] Available at https://github.com/johannesalehmann/backward-responsibility

Given a play that reached such a configuration, a natural question is to find the last state from which the game was still winnable. One can use optimistic responsibility to determine this. Consider the configuration from Figure 2a, which was the result starting with Hole 1 empty and playing $4 \to 1$, $6 \to 4$, $1 \to 6$, $12 \to 5$, $14 \to 12$, $6 \to 13$, $12 \to 14$, $15 \to 13$, $7 \to 2$, $2 \to 9$, $10 \to 8$, $13 \to 4$. A graphical depiction of the full game is given in the appendix. Computing optimistic responsibility reveals that the configuration depicted in Figure 2b is the last state with positive responsibility. If $15 \to 13$ is played in this state, all further states have responsibility 0, which tells us that the game is unwinnable. On the other hand, if $7 \to 2$, $2 \to 9$, $15 \to 6$, $6 \to 13$, $14 \to 12$, $11 \to 13$ is played from the configuration in Figure 2b, the game is won. This demonstrates that optimistic responsibility is useful when only a qualitative analysis is desired.

Example: Misrouted Train We modelled Dresden Central Station to analyse a misrouted train. The train was supposed to arrive at Platform 12 but instead arrived at Platform 13. Our goal is to determine how much responsibility each switch in the train station has for this misrouting. The relevant fragment of the station is depicted in Figure 3. The path taken by the train is indicated in bold.

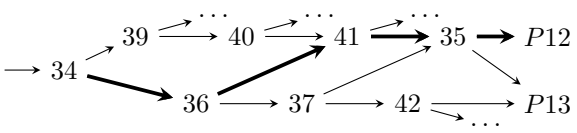


Figure 3: A train misrouted to Platform $P12$ instead of $P13$ in Dresden Central Station and the path the train took.

As we are interested in a quantitative analysis of responsibility, we choose pessimistic responsibility. This yields four responsible states: 35 has responsibility $3/4$, whereas 36, 37 and 42 have responsibility $1/12$ each. The reason for 35's high responsibility is that it can route the train to the correct track without any cooperation from other switches. On the other hand, 36, 37 and 42 need to all be in the correct position to achieve the same result. As the default assumption in pessimistic responsibility is that other switches do not cooperate, this three-way cooperation only occurs in few coalitions and therefore, the total responsibility of the three states is lower than that of 35.

Another possible route to Platform 13 is $34 \to 39 \to 40 \to 41 \to 35 \to P13$. Note however that $34, 39, 40$ and 41 have responsibility 0. This is because the route relies on 35 cooperating. As 35 can route the train correctly on its own, there is no benefit in also changing the route at the other switches.

Stochastic Algorithm

The stochastic algorithm uses the following transformation.

$$\begin{aligned}\mathcal{R}(v,s) &= \sum_{C \subseteq S \setminus \{s\}} \texttt{Weights}_{|C|}(\mathcal{R})(v(C \cup \{s\}) - v(C)) \\ &= \sum_{i=0}^{n-1} \texttt{Weights}_i(\mathcal{R}) \cdot \sum_{\substack{C \subseteq S \setminus \{s\}, \\ |C|=i}} (v(C \cup \{s\}) - v(C)).\end{aligned}$$

As each summand of the inner sum is either 0 or 1, it is sufficient to count for how many coalitions the summand is 1. The stochastic algorithm estimates this count by uniformly sampling coalitions of size i. If x out of y coalitions have value 1, then the expected count for size i is $(x/y) \cdot \binom{n}{i}$. As long as each size is sampled at least once, the algorithm is an unbiased estimator for $\mathcal{R}(v, s)$.

The number of samples for size i should be proportional to $\binom{n}{i} \cdot \texttt{Weights}_i(\mathcal{R})$. In the case of the Shapley value, each size should therefore get the same number of samples.

We have evaluated this on a collection of benchmarks. `alternating_bit` is a simple message transition protocol, `brp` is a protocol to transfer files consisting of N chunks (with *MAX* attempts), `crowds` models anonymised message routing through a group of size *CS* (with *TR* total runs), `dining_philosophers` models the well-known dining philosophers problem, `dresden_railways` models the switches in Dresden Central Station, where a train has to be routed to a specific platform and `generals` models the n-generals problem, who have to decide independently whether to attack or not.The benchmarks `brp` and `crowds` are taken from Kwiatkowska, Norman, and Parker (2012).

For each model, we have sampled for t seconds (where $t \in \{1, 10, 60\}$) and then computed the responsibility of all states using these samples. To evaluate the quality of our samples, we have repeated this procedure 20 times and determined the standard deviation of the estimated responsibility from the actual value[2]. Table 2 presents the results. The number of states of each model is given and for each run, n indicates the average number of samples and σ the standard deviation of the samples from the reference value.

Parentheses indicate insufficient coverage, which we detect by analysing the sum of the responsibilities. If it is consistently much smaller than 1, coverage is likely insufficient.

As expected, the standard deviation decreases if we sample for longer. Furthermore, bigger models show a bigger deviation than smaller models for the same sampling duration. This is to be expected as each sample takes longer for bigger models.

State Grouping

State grouping works by partitioning the set of states S into groups $G_1, \ldots, G_m$. Instead of analysing all subsets $C \subseteq S$, we instead analyse $C = \bigcup_{i \in I} G_i$ for all $I \subseteq \{1, \ldots, m\}$, i.e. if two states are in the same state group, they are either both included in the coalition or neither of them is.

[2]For the larger models, we cannot compute the exact responsibility. We therefore used the average of all runs with $t = 60$ as reference value, as this is the best approximation we have.

		$t = 1s$		$t = 10s$		$t = 60s$	
name	states	n	σ	n	σ	n	σ
`alternating_bit`	297	38.4k	0.0304	335.1k	0.0091	1844.0k	0.0038
`brp`, $N = 4$, $MAX = 2$	173	685.4k	0.0015	6963.9k	0.0005	41754.6k	0.0002
`brp`, $N = 16$, $MAX = 3$	886	71.2k	0.0087	663.5k	0.0028	3826.8k	0.0011
`crowds`, $TR = 3$, $CS = 5$	1198	49.3k	0.0120	372.3k	0.0041	2177.8k	0.0018
`dining_philosophers`, $N = 3$	36	1606.6k	0.0015	16145.9k	0.0005	95952.8k	0.0002
`dining_philosophers`, $N = 5$	393	35.5k	0.0135	321.9k	0.0049	1865.5k	0.0022
`dresden_railways`	54	1286.8k	0.0009	12537.2k	0.0003	76475.9k	0.0001
`generals`, $N = 3$	20	2055.2k	0.0017	20655.9k	0.0005	117232.9k	0.0002
`generals`, $N = 5$	112	50.8k	0.0166	603.8k	0.0049	3626.3k	0.0019
`generals`, $N = 8$	1280	(7.9k)	(0.0909)	69.9k	0.0242	313.6k	0.0104

Table 2: Evaluation of the stochastic algorithm, where n is the number of samples per run and σ is the standard deviation of the result from each run from the reference value.

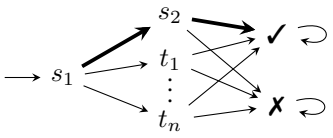


Figure 4: Railway network similar to that from Figure 1, but with multiple switches $t_1, \ldots, t_n$.

To give an intuition for state groups, recall the example from Figure 1. We can increase the complexity of the model by replacing s_3 with several switches $t_1, \ldots, t_n$, as shown in Figure 4. Running our tool for $n = 5$ reveals that each t_i has responsibility 0.0238, whereas s_1 has a responsibility of 0.3571 and s_2 has a responsibility of 0.5238 (all values rounded to four decimal places). If we do not want to analyse individual responsibility of the switches t_1 to t_5 (perhaps because they are switched by the same controller, or because we want to reduce the complexity of the model), we can use the state groups $\{s_1\}, \{s_2\}, \{t_1, \ldots, t_5\}$. With these state groups, the responsibilities are now equal to those in Table 1, i.e. $\{s_1\}$ has responsibility $1/6$, $\{s_2\}$ has responsibility $2/3$ and $\{t_1, \ldots, t_5\}$ has responsibility $1/6$.

In addition to simplifying the results, state groups also decrease runtime, as the algorithm is exponential only in the number of *groups*, but linear in the number of states.

Example: Dining Philosophers Consider an implementation of the dining philosophers with a simple scheduler. For four philosophers, this model has 180 states, which cannot be analysed with the exact algorithm. If we group the states by whose turn it is, we can compute responsibility values in less than a millisecond (and it is revealed that, in our case, all philosophers are equally responsible for the deadlock).

Example: BRP It is also possible to combine the stochastic algorithm with state grouping. For example, consider the `brp` model from Table 2. If we increase the parameters to $N = 64, MAX = 5$, the model has 5192 states and even 60s of sampling result in insufficient coverage for individual responsibility (as the sum of responsibilities is consistently below 1). If we instead group states by which chunk is being processed, we get 64 groups and after running for 1s, the tool is able to approximate the responsibility values with a standard deviation of 0.0006.

6 Conclusion

We have presented two notions of backward responsibility. For optimistic responsibility, every state with positive responsibility has the same amount of responsibility. The characterisation provides a simple test to find the states with positive responsibility by computing whether they can change the outcome by themselves. While this makes computation straight-forward, it also means that it does not rank states by responsibility in the way pessimistic responsibility does.

Pessimistic responsibility, on the other hand, is harder to compute, but can give positive responsibility both to states on the counterexample and to states that are not. Furthermore, not all responsible states have the same responsibility.

We have demonstrated that our technique works in practice and that a stochastic algorithm can be used to analyse much larger models than would otherwise be feasible.

Future Work: Our investigation was restricted to safety games. As our definitions can readily be adapted to other classes of games, such as Büchi games, it is of interest to determine whether our results still hold for these classes and to analyse the complexity of the algorithms for them.

Another avenue for future work is the presentation of the data to the user. Our tool gives the responsibility values for each state. To facilitate debugging, it would be useful to take this state-based responsibility and map it back to the specification language (in our case, this is the PRISM language). However, such a mapping is non-trivial, as there is no direct correspondence between states and source-code lines.

Acknowledgements

Funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden, by DFG Grant 389792660 as part of TRR 248 (Foundations of Perspicuous Software Systems) and by BMBF (Federal Ministry of Education and Research) in DAAD project 57616814 (SECAI, School of Embedded Composite AI) as part of the program Konrad Zuse Schools of Excellence in Artificial Intelligence.

References

Baier, C.; Funke, F.; and Majumdar, R. 2021. A Game-Theoretic Account of Responsibility Allocation. In *IJCAI*, 1773–1779. International Joint Conferences on Artificial Intelligence Organization.

Baier, C.; and Katoen, J. 2008. *Principles of model checking*. MIT Press.

Baier, C.; van den Bossche, R.; Klüppelholz, S.; Lehmann, J.; and Piribauer, J. 2024. Backward Responsibility in Transition Systems Using General Power Indices. *arXiv preprint*.

Ball, T.; Naik, M.; and Rajamani, S. K. 2003. From symptom to cause: localizing errors in counterexample traces. In *Proceedings of the 30th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 97–105. Association for Computing Machinery.

Beer, I.; Ben-David, S.; Chockler, H.; Orni, A.; and Trefler, R. 2012. Explaining counterexamples using causality. *Formal Methods in System Design*, 40: 20–40.

Bell, G. I. 2007. Solving triangular peg solitaire. *arXiv preprint math/0703865*.

Coenen, N.; Finkbeiner, B.; Frenkel, H.; Hahn, C.; Metzger, N.; and Siber, J. 2022. Temporal causality in reactive systems. In *International Symposium on Automated Technology for Verification and Analysis*, 208–224. Springer.

Dubey, P.; Neyman, A.; and Weber, R. J. 1981. Value Theory without Efficiency. *Mathematics of Operations Research*, 6(1): 122–128.

Friedenberg, M.; and Halpern, J. Y. 2019. Blameworthiness in multi-agent settings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 525–532.

Grädel, E.; Thomas, W.; and Wilke, T., eds. 2002. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer.

Groce, A.; Chaki, S.; Kroening, D.; and Strichman, O. 2006. Error explanation with distance metrics. *International Journal on Software Tools for Technology Transfer*, 8: 229–247.

Halpern, J.; and Kleiman-Weiner, M. 2018. Towards formal definitions of blameworthiness, intention, and moral responsibility. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Kwiatkowska, M.; Norman, G.; and Parker, D. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In Gopalakrishnan, G.; and Qadeer, S., eds., *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, 585–591. Springer.

Kwiatkowska, M. Z.; Norman, G.; and Parker, D. 2012. The PRISM Benchmark Suite. In *QEST*, 203–204. IEEE Computer Society.

Laruelle, A.; and Valenciano, F. 2001. Shapley-Shubik and Banzhaf Indices Revisited. *Mathematics of Operations Research*, 26(1): 89–104.

Lehmann, J. 2024. Tool to compute backward responsibility in transition systems using general power indices.

Leitner-Fischer, F.; and Leue, S. 2013. Causality checking for complex system models. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, 248–267. Springer.

Mascle, C.; Baier, C.; Funke, F.; Jantsch, S.; and Kiefer, S. 2021. Responsibility and verification: Importance value in temporal logics. In *LICS*, 1–14. IEEE.

Renieres, M.; and Reiss, S. P. 2003. Fault localization with nearest neighbor queries. In *18th IEEE International Conference on Automated Software Engineering, 2003. Proceedings.*, 30–39. IEEE.

Seshia, S. A.; Sadigh, D.; and Sastry, S. S. 2022. Toward verified artificial intelligence. *Communications of the ACM*, 65(7): 46–55.

Shapley, L. S.; et al. 1953. A value for n-person games. *Contributions to the Theory of Games*, 2.

Van de Poel, I. 2011. The relation between forward-looking and backward-looking responsibility. In *Moral responsibility: Beyond free will and determinism*, 37–52. Springer.

Wang, C.; Yang, Z.; Ivančić, F.; and Gupta, A. 2006. Whodunit? causal analysis for counterexamples. In *International Symposium on Automated Technology for Verification and Analysis*, 82–95. Springer.

Zeller, A. 2002. Isolating cause-effect chains from computer programs. In *SIGSOFT FSE*, 1–10. ACM.