Abstract Interpretation of Fixpoint Iterators with Applications to Neural Networks

MARK NIKLAS MÜLLER, MARC FISCHER, ROBIN STAAB, and MARTIN VECHEV,
Department of Computer Science, ETH Zurich, Switzerland

We present a new abstract interpretation framework for the precise over-approximation of numerical fixpoint iterators.Our key observation is that unlike in standard abstract interpretation (AI), typically used to over-approximate *all* reachable program states, in this setting, one only needs to abstract the concrete fixpoints, i.e., the *final* program states. Our framework targets numerical fixpoint iterators with convergence and uniqueness guarantees in the concrete and is based on two major technical contributions: (i) theoretical insights which allow us to compute sound and precise fixpoint abstractions *without using joins*, and (ii) a new abstract domain, CH-Zonotope, which admits efficient propagation and inclusion checks while retaining high precision.

We implement our framework in a tool called Craft and evaluate it on a novel fixpoint-based neural network architecture (monDEQ) that is particularly challenging to verify. Our extensive evaluation demonstrates that Craft exceeds the state-of-the-art performance in terms of speed (two orders of magnitude), scalability (one order of magnitude), and precision (25% higher certified accuracies).

CCS Concepts: • **Theory of computation → Abstraction**; **Program verification**; • **Computing methodologies → Neural networks**.

Additional Key Words and Phrases: fixpoint, abstract interpretation, equlibrium models, adversarial robustness

ACM Reference Format:
Mark Niklas Müller, Marc Fischer, Robin Staab, and Martin Vechev. 2023. Abstract Interpretation of Fixpoint Iterators with Applications to Neural Networks. *Proc. ACM Program. Lang.* 7, PLDI, Article 138 (June 2023), 25 pages. https://doi.org/10.1145/3591252

1 INTRODUCTION

Abstract interpretation (AI) [Cousot and Cousot 1977a, 1992a] is a popular static analysis technique, typically used to over-approximate all reachable states of a given program for a particular set of (potentially infinite) concrete inputs, captured by a pre-condition. Given an *abstract domain* for representing abstract program states and *abstract transformers* for capturing the effects of program statements on these abstract states, AI operates by starting with the pre-condition and applying the abstract transformers corresponding to each program statement until a so-called abstract (post-)fixpoint is reached, i.e., any further application of the abstract transformers does not change the computed abstraction. Under reasonable assumptions and in the absence of unbounded loops, this approach is guaranteed to terminate with a sound abstraction of *all* — intermediate and final — program states. To handle unbounded loops, special techniques such as Kleene iteration with widening and narrowing [Cousot and Cousot 1992b] are required to ensure termination.

Authors' address: Mark Niklas Müller, mark.mueller@inf.ethz.ch; Marc Fischer, marc.fischer@inf.ethz.ch; Robin Staab, robin.staab@inf.ethz.ch; Martin Vechev, martin.vechev@inf.ethz.ch,
Department of Computer Science, ETH Zurich, Universitätsstrasse 6, 8092, Zurich, Switzerland.

2475-1421/2023/6-ART138
https://doi.org/10.1145/3591252

Interestingly, for an important class of programs with unbounded loops that themselves compute (concrete) fixpoints, e.g., numerical solvers, typically, only the resulting concrete fixpoints, i.e. the *final state* of the concrete program rather than the intermediate states, are of interest. Using Kleene iteration in this setting, even with exact joins, leads to abstractions that include the union over all iteration states, making them inherently imprecise. A desirable goal then is to develop an abstract interpretation approach that targets only the precise abstraction of these final states.

This Work: Abstract Interpretation of Fixpoint Iterators. In this work, we introduce the first abstract interpretation framework, focusing on fixpoint iterators that possess convergence guarantees in the concrete. Our framework is based on two major contributions: (i) we present new theoretical insights which allow us to compute sound and precise fixpoint abstractions *without using joins*. That is, we do not require Kleene iteration, typically used in AI to handle unbounded loops [Gange et al. 2013; Putot 2012], and further demonstrate that Kleene iteration is unsuitable for our class of programs. In addition, these insights enable us to further tighten the obtained abstractions by leveraging the convergence properties of the abstracted fixpoint iterator. While our method can be instantiated with any abstract domain, (ii) we introduce a novel abstract domain, called CH-Zonotope, based on the Zonotope abstraction [Ghorbal et al. 2009; Singh et al. 2018], combined with the notion of order-reduction [Kopetzki et al. 2017; Yang and Scott 2018]. Unlike Zonotope, our domain ensures constant representation size and allows for efficient yet precise inclusion checks – only $O(p^3)$ instead of $O(p^6)$ in the dimension p – critical for handling fixpoint iterations.

We implement our framework in a tool called Craft and demonstrate its effectiveness on the robustness verification of monDEQs (Monotone Operator Deep Equilibrium Models) [Winston and Kolter 2020], a novel fixpoint-based neural architecture combining high-dimensionality and highly non-linear iterations, thus representing a particularly challenging class of fixpoint iterators. We remark that Craft can serve as a basis for future investigations of other fixpoint-based neural architectures such as stiff neural ODEs [Kim et al. 2021] or SatNets [Wang et al. 2019].

Main Contributions. Our core contributions are:

- A domain-specific abstract interpretation framework for high-dimensional fixpoint iterators with convergence guarantees in the concrete. (Section 3).
- CH-Zonotope, a novel abstract domain that enables both efficient abstract fixpoint iteration and inclusion checks (Section 4).
- Craft, a complete implementation of our framework and abstract domain (Section 5.2).
- An extensive evaluation demonstrating the effectiveness of CH-Zonotope and showing that Craft achieves a new state-of-the-art for monDEQ verification.

2 OVERVIEW

We now elaborate on the key challenges of abstracting fixpoint iterators and our approach to overcoming these. As a running example, we use monDEQs, a novel neural architecture based on high-dimensional fixpoint iterations and an instance of the class of programs we target. Thus, we begin with a short background on neural networks and their analysis.

Neural Network Verification. Given a neural network $\boldsymbol{h}\colon \mathbb{R}^{d_{in}} \mapsto \mathbb{R}^r$, a precondition $\varphi(\boldsymbol{x})$, and postcondition $\psi(\boldsymbol{h}(\boldsymbol{x}))$, the goal of neural network verification is to show that $\varphi(\boldsymbol{x}) \models \psi(\boldsymbol{h}(\boldsymbol{x}))$. To this end, we construct a sound verifier to show $\varphi(\boldsymbol{x}) \vdash \psi(\boldsymbol{h}(\boldsymbol{x}))$, i.e., that $\psi(\boldsymbol{h}(\boldsymbol{x}))$ can be derived from $\varphi(\boldsymbol{x})$, implying by the soundness of the verifier $\varphi(\boldsymbol{x}) \models \psi(\boldsymbol{h}(\boldsymbol{x}))$, i.e., that $\varphi(\boldsymbol{x})$ entails $\psi(\boldsymbol{h}(\boldsymbol{x}))$.

A common instantiation of this problem is found in image classification. There, $\boldsymbol{x}$ is an image, $\boldsymbol{h}$ an image classifier, $\varphi(\boldsymbol{x})$ an ℓ_p-norm-ball around $\boldsymbol{x}$, e.g., $\varphi(\boldsymbol{x}) := \{\boldsymbol{x}' \in \mathbb{R}^{d_{in}} \mid \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \leq \epsilon\}$, $\psi(\boldsymbol{h}(\boldsymbol{x}))$ denotes classification to the correct class, and showing $\varphi(\boldsymbol{x}) \models \psi(\boldsymbol{h}(\boldsymbol{x}))$ formally verifies robustness to adversarial examples [Goodfellow et al. 2015; Szegedy et al. 2014].

A popular approach to constructing neural network verifiers is to adapt abstract interpretation techniques to handle hundreds of thousands of variables [Gehr et al. 2018; Singh et al. 2018, 2019b]. There, the precondition $\varphi(\boldsymbol{x})$ is encoded as an abstract element and propagated through the network layer-by-layer using abstract transformers before the resulting abstraction of the network output is checked against the postcondition $\psi(\boldsymbol{h}(\boldsymbol{x}))$.

Fixpoint-based Neural Networks. Neural architectures based on fixpoint computations such as monDEQs (formally discussed in Section 5), however, do not simply apply a fixed number of layers, instead iteratively applying an iterator in an unbounded loop until a fixpoint is reached. We highlight this difference in Fig. 1, where we contrast pseudocode for a standard feed-forward neural network (left) and a monDEQ (right). Let us consider an example monDEQ classifier $\boldsymbol{h}\colon [-1,1]^2 \mapsto \{0,1\}$:

```
def NN(x):
s_1 = layer_1(x)
⋮
s_{L-1} = layer_{L-1}(s_{L-2})
y = layer_L(s_{L-1})
return y
```

```
def monDEQ(x):
  s_0 = 0, i = 0
  while not converged(s_i):
    i = i + 1
    s_i = g(x, s_{i-1})
  y = layer_y(s_i)
  return y
```

Fig. 1. Pseudocode for a standard neural network (left) and a monDEQ (right).

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{s}) \coloneqq ReLU\left(\tfrac{1}{10}\left(\begin{smallmatrix}5 & -1\\ 1 & 5\end{smallmatrix}\right)\boldsymbol{s} + \tfrac{1}{10}\left(\begin{smallmatrix}1 & 1\\ -1 & 1\end{smallmatrix}\right)\boldsymbol{x}\right) \qquad \text{layer}_{\boldsymbol{y}}(\boldsymbol{s}) \coloneqq \left(\begin{smallmatrix}1 & -1\end{smallmatrix}\right)\boldsymbol{s}, \tag{1}$$

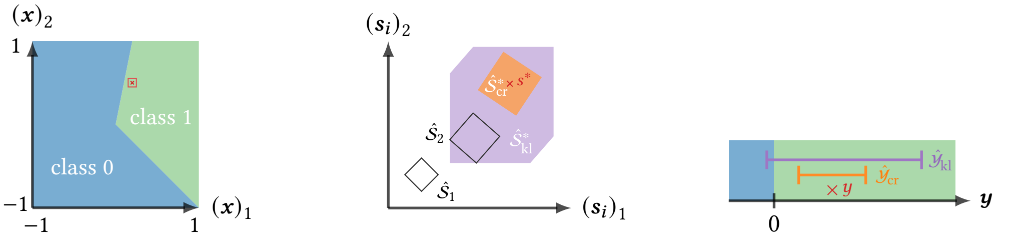
returning class 1 if $\boldsymbol{y}(\boldsymbol{s}^*) \coloneqq \boldsymbol{h}(\boldsymbol{x}) = \text{layer}_{\boldsymbol{y}}(\boldsymbol{s}^*) > 0$ and else class 0, where $\boldsymbol{s}^* = \boldsymbol{g}(\boldsymbol{x},\boldsymbol{s}^*)$ denotes the fixpoint found by iterating $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{s})$. Given an example input $\boldsymbol{x} \coloneqq \left(\begin{smallmatrix}0.2\\ 0.5\end{smallmatrix}\right) = \frac{1}{10}\left(\begin{smallmatrix}2\\ 5\end{smallmatrix}\right)$, we compute the fixpoint $\boldsymbol{s}^*$ by iteratively applying $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{s}_i)$. We initialize the iteration with $\boldsymbol{s}_0 = \left(\begin{smallmatrix}0\\ 0\end{smallmatrix}\right)$ and obtain

$$\begin{aligned}
\boldsymbol{s}_{i+1} &\coloneqq \boldsymbol{g}(\boldsymbol{x},\boldsymbol{s}_i) = ReLU\left(\tfrac{1}{10}\left(\begin{smallmatrix}5 & -1\\ 1 & 5\end{smallmatrix}\right)\boldsymbol{s}_i + \tfrac{1}{100}\left(\begin{smallmatrix}1 & 1\\ -1 & 1\end{smallmatrix}\right)\left(\begin{smallmatrix}2\\ 5\end{smallmatrix}\right)\right) = ReLU\left(\tfrac{1}{10}\left(\begin{smallmatrix}5 & -1\\ 1 & 5\end{smallmatrix}\right)\boldsymbol{s}_i + \tfrac{1}{100}\left(\begin{smallmatrix}7\\ 3\end{smallmatrix}\right)\right)\\
\boldsymbol{s}_1 &\coloneqq ReLU\left(\tfrac{1}{10}\left(\begin{smallmatrix}0\\ 0\end{smallmatrix}\right) + \tfrac{1}{100}\left(\begin{smallmatrix}7\\ 3\end{smallmatrix}\right)\right) = \tfrac{1}{100}\left(\begin{smallmatrix}7\\ 3\end{smallmatrix}\right) && \|\boldsymbol{s}_1 - \boldsymbol{s}_0\| = 0.0762\\
\boldsymbol{s}_2 &\coloneqq ReLU\left(\tfrac{1}{1000}\left(\begin{smallmatrix}32\\ 22\end{smallmatrix}\right) + \tfrac{1}{100}\left(\begin{smallmatrix}7\\ 3\end{smallmatrix}\right)\right) = \tfrac{1}{1000}\left(\begin{smallmatrix}102\\ 52\end{smallmatrix}\right) && \|\boldsymbol{s}_2 - \boldsymbol{s}_1\| = 0.0389\\
\boldsymbol{s}^* &\approx \left(\begin{smallmatrix}0.1231\\ 0.0846\end{smallmatrix}\right).
\end{aligned}$$

We observe that the residual $\|\boldsymbol{s}_{i+1} - \boldsymbol{s}_i\|$ decreases quickly as we converge towards the fixpoint $\boldsymbol{s}^*$ and note that this convergence to a unique fixpoint is guaranteed for monDEQs [Winston and Kolter 2020]. We can thus compute the fixpoint $\boldsymbol{s}^*$ to arbitrary precision, only depending on the termination condition, `converged(`s_i`)` in Fig. 1. In our example, we obtain $\boldsymbol{y}(\boldsymbol{s}^*) \approx 0.0385 > 0$ and thus return class 1. In Fig. 2, we illustrate this inference process, showing the decision landscape of $\boldsymbol{h}$ on $[-1,1]^2$ (Fig. 2a), the obtained fixpoints (Fig. 2b), and the resulting output (Fig. 2c). We highlight the points corresponding to our example input $\boldsymbol{x}$ with a red × and will explain the orange and purple regions shortly. In the following examples, we assume that `converged(`s_i`)` is chosen such that we reach the true fixpoints up to machine precision.

2.1 Abstract Interpretation for Fixpoint Iterators: A Motivation

While the construction of abstract interpretation based verifiers for loop-free programs such as feed-forward networks (left in Fig. 1) is conceptually straightforward, fixpoint iterators such as monDEQs present a greater challenge due to their unbounded loops (right in Fig. 1). To motivate the need for a domain-specific abstraction framework, we will first illustrate that generic abstract interpretation techniques are inherently not suitable for this task due to three fundamental reasons: (i) the analysis of fixpoint-iterators requires only the *last* iteration state, containing the concrete fixpoints, instead of all intermediate iteration states, to be abstracted, (ii) while in general abstract interpretation, an abstract transformer of the termination condition has to be evaluated to refine the obtained abstract state, fixpoint iterators allow the mathematical invariants that are enforced by the termination condition to be leveraged directly, leading to much more precise results, and finally, (iii) standard techniques do not take advantage of the key convergence properties of the concrete fixpoint iterator $\boldsymbol{g}$, which we leverage in order to drastically improve precision.



(a) Classification of concrete inputs in $[-1, 1]^2$ by $\boldsymbol{h}$.

(b) Abstractions of iteration steps $\boldsymbol{s}_i$.

(c) Abstractions of the output $\boldsymbol{y}$ and resulting classification.

Fig. 2. Example Visualization: The concrete input $\boldsymbol{x}$ (red ×) yields the fixpoint $\boldsymbol{s}^*(\boldsymbol{x})$ (×), and prediction $y(\boldsymbol{x})$ (×). Propagating the input region $\mathcal{X}$ (red □) with Kleene iteration and Craft yields the abstract iteration steps $\hat{\mathcal{S}}_1$ and $\hat{\mathcal{S}}_2$ and finally the fixpoint $\hat{\mathcal{S}}^*_{\mathrm{kl}}$ and $\hat{\mathcal{S}}^*_{\mathrm{cr}}$ and corresponding output abstraction $\hat{\mathcal{Y}}_{\mathrm{kl}}$ and $\hat{\mathcal{Y}}_{\mathrm{cr}}$, respectively.

2.2 Challenge: Precise Loop Abstraction

Abstract Interpretation (AI) [Cousot and Cousot 1977a, 1979, 1992a] is an analysis technique that allows reasoning over the behavior of programs for sets of inputs. Conceptually, a set of program inputs, e.g., those specified by the precondition φ, is represented symbolically and then propagated through the program to determine whether the postcondition ψ is satisfied for all these inputs.

Formally, we over-approximate sets of concrete inputs from domain C with abstract elements from an abstract domain $\mathcal{A}$. To retrieve the set of concrete values $\gamma(\hat{\mathcal{S}})$ represented by an abstract element $\hat{\mathcal{S}} \in \mathcal{A}$, we define the concretization function $\gamma : \mathcal{A} \mapsto \wp(C)$. Equipped with a partial order $\sqsubseteq$, an abstract domain forms a poset such that $\hat{\mathcal{S}}_1 \sqsubseteq \hat{\mathcal{S}}_2 \implies \gamma(\hat{\mathcal{S}}_1) \subseteq \gamma(\hat{\mathcal{S}}_2)$. This allows us to define the (quasi [Gange et al. 2013]) join $\hat{\mathcal{S}}_1 \sqcup \hat{\mathcal{S}}_2$ of two abstract elements $\hat{\mathcal{S}}_1, \hat{\mathcal{S}}_2$ as their least (any) upper bound with respect to $\sqsubseteq$. Importantly, if the join exists, we have $\gamma(\hat{\mathcal{S}}_1) \cup \gamma(\hat{\mathcal{S}}_2) \subseteq \gamma(\hat{\mathcal{S}}_1 \sqcup \hat{\mathcal{S}}_2)$. We capture the effect of a concrete function $f : C \to C$, e.g., a program statement, on an abstract element $\hat{\mathcal{S}} \in \mathcal{A}$, using a sound abstract transformer $f^\# : \mathcal{A} \to \mathcal{A}$ such that $\forall s \in \gamma(\hat{\mathcal{S}}),\ f(s) \in \gamma(f^\#(\hat{\mathcal{S}}))$.

Standard Abstract Post-Fixpoint Computation. In AI, there are generally two possible outcomes when control flow, such as the if-statement in Fig. 3, is encountered: (i) either the abstract state allows to show that the same branch is taken for all abstracted values and we only have to consider the abstract transformer of that branch, e.g., $f_a^\#$ to obtain $\hat{\mathcal{Y}} = f_a^\#(\hat{\mathcal{X}})$, or (ii) if we can not rule out either branch, both branches have to be considered and we obtain the join of the resulting abstract states $\hat{\mathcal{Y}} = f_a^\#(\hat{\mathcal{X}}) \sqcup f_b^\#(\hat{\mathcal{X}})$. Most control flow, including bounded loops, can be handled in this way.

```
x = ...
if check(x):
  y = f_a(x)
else:
  y = f_b(x)
return y
```

Fig. 3. Example branching behavior.

Unbounded loops, such as those encountered in monDEQs, however, present a special challenge as the above approach will typically not terminate. A common solution to this problem is the so-called *Kleene iteration*. For a loop of the form `while condition(s): s = f(s)`, Kleene iteration computes $\hat{\mathcal{S}}_i = \hat{\mathcal{S}}_{i-1} \sqcup f^\#(\hat{\mathcal{S}}_{i-1})$ until convergence or formally until an order-theoretic (post)-fixpoint $\hat{\mathcal{S}}^* := \hat{\mathcal{S}}_{i-1} \sqsupseteq \hat{\mathcal{S}}_i$ is reached. In practice, widening [Cousot and Cousot 1992b] is often required for Kleene iteration to terminate. Unfortunately, the obtained precision is heavily dependent on the existence of precise abstract transformers for the termination condition. If we lack such transformers, e.g., due to complex, non-linear, non-convex termination conditions, the obtained abstraction is often imprecise. To recover some precision, we apply *semantic unrolling* [Blanchet et al. 2002], i.e., unroll the first k loop iterations for which we can show that the termination condition is not satisfied and thus iterate $\hat{\mathcal{S}}_i = f^\#(\hat{\mathcal{S}}_{i-1})$ for $i \leq k$, avoiding the join $\hat{\mathcal{S}}_i = \hat{\mathcal{S}}_{i-1} \sqcup f^\#(\hat{\mathcal{S}}_{i-1})$.

Example (cont.) Let us apply Kleene iteration to our example to illustrate these imprecision issues. For the monDEQ in Eq. (1), we let $\mathcal{X} = \varphi(\boldsymbol{x}) = \{\boldsymbol{x} + \delta \mid \delta \in \mathbb{R}^2, \|\delta\|_\infty \leq 0.05\}$ denote a small region around $\boldsymbol{x}$ (red □ in Fig. 2a) and ψ the classification to class 1. We initialize $\hat{\mathcal{S}}_0$ such that $\gamma(\hat{\mathcal{S}}_0) = \{\mathbf{0}\}$ and apply Kleene iteration with semantic unrolling ($k = 2$) to the abstraction $\boldsymbol{g}^\#$ of the iterator $\boldsymbol{g}$ (Eq. (1)) using the Zonotope domain [Singh et al. 2018] to compute an abstract post-fixpoint of the loop. We illustrate the intermediate states $\hat{\mathcal{S}}_1$ and $\hat{\mathcal{S}}_2$ as well as the final fixpoint $\hat{\mathcal{S}}^*_{\text{kl}}$ (purple) in Fig. 2b. Note how the second state $\hat{\mathcal{S}}_2$ is included in the post-fixpoint $\hat{\mathcal{S}}^*_{\text{kl}}$. Applying the classification layer, we obtain $\hat{\mathcal{Y}}_{\text{kl}} = \boldsymbol{y}^\#(\hat{\mathcal{S}}^*_{\text{kl}})$ (purple interval in Fig. 2c). As the interval contains 0, we are unable to verify the postcondition that all points in $\mathcal{X}$ get classified to class 1 (even though they do).

Domain-Specific Abstraction of Fixpoint Iterators. We now propose a domain-specific approach targeting the abstraction of fixpoint iterators. Recall that in Section 2.1 we discussed three reasons for the imprecision of the standard approach, which we address as follows. First, in our setting, we note that it is sufficient to abstract the set containing all concrete fixpoints $\mathcal{S}^* = \{\boldsymbol{s}^* \mid \boldsymbol{s}^* = g(\boldsymbol{x}, \boldsymbol{s}^*), \boldsymbol{x} \in \mathcal{X}\}$ instead of the union of all iteration states arriving at the loop head. Second, instead of requiring an abstract transformer for the termination condition $\texttt{converged}(s_i)$, we leverage the mathematical invariants enforced by the condition, namely that a fixpoint has been reached, to show that it suffices to iterate $\hat{\mathcal{S}}_i = \boldsymbol{g}^\#(\hat{\mathcal{X}}, \hat{\mathcal{S}}_{i-1})$ – *without the use of joins* and *without requiring abstract transformers for the termination condition* – until we reach containment ($\hat{\mathcal{S}}_{i-1} \sqsupseteq \hat{\mathcal{S}}_i$), to guarantee that $\gamma(\hat{\mathcal{S}}_i)$ contains all concrete fixpoints $\mathcal{S}^*$. Finally, we leverage the convergence properties of $\boldsymbol{g}$ and prove that an additional $j > 0$ applications of $\boldsymbol{g}^\#$ to $\hat{\mathcal{S}}_i$ yield abstractions which may not be included in $\hat{\mathcal{S}}_i$, yet are always sound, i.e., contain all concrete fixpoints $\mathcal{S}^*$, while being empirically much tighter than $\hat{\mathcal{S}}_i$. We refer to this property of $\boldsymbol{g}^\#$ as *fixpoint set preservation.* Finally, we leverage these insights to perform precise verification of monDEQs by computing $\hat{\mathcal{Y}}$ from the resulting over-approximation $\hat{\mathcal{S}}_{i+j}$ and then checking $\psi(\hat{\mathcal{Y}})$.

Example (cont.) Let us continue our example: We initialize our iteration as for Kleene but iterate $\hat{\mathcal{S}}_i = \boldsymbol{g}^\#(\hat{\mathcal{X}}, \hat{\mathcal{S}}_{i-1})$ until we find $\hat{\mathcal{S}}_i \sqsubseteq \hat{\mathcal{S}}_{i-1}$, again visualizing the iteration in Fig. 2b. After sharing $\hat{\mathcal{S}}_1$ and $\hat{\mathcal{S}}_2$ with the (unrolled) Kleene iteration, we, in contrast to Kleene iteration, do not have to compute the join over iteration states and thus reach the much more precise abstraction $\hat{\mathcal{S}}^*_{\text{cr}}$ (orange). Indeed, applying the last layer yields a much more precise $\hat{\mathcal{Y}}_{\text{cr}} = \boldsymbol{y}^\#(\hat{\mathcal{S}}^*_{\text{cr}})$, allowing us to show that $y > 0$ and thus certify that all inputs in the red region are indeed classified to class 1.

Summary: Domain-Specific vs. Standard Abstractions. To summarize, as our setting is motivated by computing fixpoint set over-approximations, we are only interested in the *final* state obtained by the iterator and not in the intermediate program states. This difference in the objective combined with the mathematical properties of the iterator and termination condition allows our domain-specific approach to compute much tighter fixpoint approximations than standard AI.

2.3 Challenge: Efficient Computation

Choice of Abstract Domain. While our abstraction framework for fixpoint iterators is in itself a compelling result, it comes with another challenge, namely, the need for an abstract domain that satisfies all of the following criteria: (i) efficient propagation through the iterator $\boldsymbol{g}$, (ii) efficient inclusion checks in high dimensions, *and* (iii) high precision.

Table 1. Comparison of CH-Zonotope to other abstract domains, for fixpoint abstraction.

	Iteration	Inclusion	Precision
Box	✓	✓	✗
(Hybrid) Zonotope	✗	✗	✓
Polyhedra	✗	✗	?
CH-Zonotope	✓	✓	✓

To motivate this challenge, we first discuss why existing abstract domains used in neural network verification, shown in Table 1, are unable to satisfy these criteria. Consider L iterations of the iterator $\boldsymbol{g}$ with a latent variable $\boldsymbol{s} \in \mathbb{R}^p$ of dimension p.

The *Box* abstraction [Gehr et al. 2018; Gowal et al. 2018; Mirman et al. 2018] is the simplest commonly used abstract domain. Due to its constant-size representation, it can be efficiently propagated ($\mathcal{O}(Lp^2)$) and permits fast $\mathcal{O}(p)$ inclusion checks. However, as demonstrated in Section 6.4, it loses too much precision to be practically effective.

(Hybrid) Zonotopes [Gehr et al. 2018; Mirman et al. 2018; Singh et al. 2018; Wong and Kolter 2018] allow for more precision, at the cost of a growing representation size, increasing the propagation cost to $\mathcal{O}(L^2p^3)$. Further, exact inclusion checks are known to be co-NP-complete [Kulmburg and Althoff 2021] and even approximate ones [Sadraddini and Tedrake 2019] (between $\mathcal{O}(p^6)$ and $\mathcal{O}(L^3p^6)$) become intractable in high ($\gtrsim 50$) dimensions.

(Restricted) Polyhedra based methods that propagate linear bounds [Singh et al. 2019b; Zhang et al. 2018] are state-of-the-art for applications where runtime is critical [Serre et al. 2021]. These methods are typically more precise than Zonotope and have identical time complexity ($\mathcal{O}(L^2p^3)$). However, as they yield polyhedra in the input-output space of the abstracted program, the input dimensions have to first be projected out to perform inclusion checks in the output space. While the inclusion checks themselves have polynomial complexity [Sadraddini and Tedrake 2019], the projection step is co-NP-hard [Kellner 2015], making the overall check intractable.

The CH-Zonotope Domain. To address the above challenges, we introduce the CH-Zonotope domain in Section 4. It builds on Hybrid Zontopes [Mirman et al. 2018], allows for an efficient inclusion check ($\mathcal{O}(p^3)$), and, thanks to the strategic use of order reduction [Kopetzki et al. 2017], ensures constant representation size, allowing for fast and efficient propagation ($\mathcal{O}(Lp^3)$).

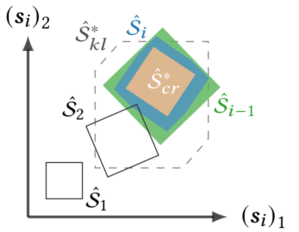


Fig. 4. Analyzing our running example with the CH-Zonotope domain.

Example (cont.) We now use the CH-Zonotope domain to analyze our running example and illustrate the result in Fig. 4. We regularly apply order reduction (discussed later) to the intermediate CH-Zonotope to limit its representation size and thus obtain different, simpler intermediate states than with Zonotope. We find a post-fixpoint of the iterator when our efficient inclusion check (also discussed later) shows that the blue region $\hat{\mathcal{S}}_i$ is contained within the green one ($\hat{\mathcal{S}}_{i-1}$). While the blue $\hat{\mathcal{S}}_i$ is thus an abstraction of the true fixpoint set, it is still relatively loose. Leveraging fixpoint set preservation (as discussed earlier) and applying additional abstract iterations $\boldsymbol{g}^\#$, we obtain the much tighter fixpoint set abstraction $\hat{\mathcal{S}}^*_{\text{cr}}$ (orange). It is almost identical to the one obtained with the much more expensive analysis based on standard Zonotope (see Fig. 2b), infeasible in higher dimensions, and much more precise than the one obtained with Kleene iteration ($\hat{\mathcal{S}}^*_{kl}$ shown dashed).

2.4 The Craft Verifier

Combining our theoretical insights and the CH-Zonotope domain, we introduce Craft, an efficient verifier of high-dimensional fixpoint iterations with convergence guarantees, based on the abstract fixpoint iterations outlined above. We discuss Craft in detail in Section 5.2, providing soundness proofs and detailed engineering considerations, before demonstrating in an extensive evaluation that it achieves state-of-the-art performance for monDEQ verification in Section 6.

3 ABSTRACTING FIXPOINT ITERATIONS

In this section, we propose a novel, domain-specific abstract interpretation approach for (high-dimensional) fixpoint iterations.

Fixpoint Iterations. We consider the general case of a function $\boldsymbol{f}$ with a unique fixpoint $\boldsymbol{z}^*(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}^*)$ given a bounded input $\boldsymbol{x}$, i.e. $\|\boldsymbol{x}\| < \infty$. Allowing for preprocessing on $\boldsymbol{x}$ and postprocessing on $\boldsymbol{z}^*$, this encompasses a wide range of problems including monDEQs.

Fixpoint solvers. Often, iteratively applying $\boldsymbol{f}$ converges only slowly or not at all towards a fixpoint [Winston and Kolter 2020]. Instead, iterative root-finding algorithms are applied to $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}) - \boldsymbol{z}$ to find the fixpoint [Bai et al. 2019]. We will introduce specific instantiations later (see Section 5) and for now assume that we have access to a so-called *fixpoint solver* $\boldsymbol{g}_\alpha$ with parameters α which converges to a unique fixpoint in finitely many steps, i.e., $\forall \epsilon \in \mathbb{R}^{>0}, \exists l \in \mathbb{N}, \forall k > l, \|z_k(\boldsymbol{x}) - \boldsymbol{z}^*(\boldsymbol{x})\| < \epsilon$.

```
def fixpoint(x):
  z = 0
  u = 0
  while not converged(z):
    z, u = g_α(x, z, u)
  return z
```

Fig. 5. Iterative fixpoint computation with solver $\boldsymbol{g}_\alpha(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u})$.

Concrete Semantics. We write $\boldsymbol{g}_\alpha(\boldsymbol{x}, \boldsymbol{s}_n)$ for the concrete semantics of one iteration of a generic fixpoint solver, where the latent variable $\boldsymbol{s} \longrightarrow [\boldsymbol{z}; \boldsymbol{u}]$ contains an auxiliary variable $\boldsymbol{u}$ in addition to $\boldsymbol{z}$. We build the concrete semantics for specific instantiations of $\boldsymbol{g}_\alpha$ directly from those of the constituting mathematical operations, e.g., in Python [Guth 2013]. We write $\Xi(\boldsymbol{x})$ for the concrete semantics of the fixpoint solver iterated until convergence (i.e., with $\epsilon \longrightarrow 0$) given an initialization of $\boldsymbol{s}_0 = \boldsymbol{0}$, as illustrated in the pseudo-code for a fixpoint solver shown in Fig. 5. We construct these concrete semantics from those of any $\boldsymbol{g}_\alpha$ satisfying the above convergence guarantees.

Abstract Semantics. We let $\mathcal{X} \subseteq \mathbb{R}^q$ denote a set of inputs, $\mathcal{S}_n \subseteq \mathbb{R}^p$ the corresponding intermediate solver states at step n (we write $[\mathcal{Z}_n, \mathcal{U}_n] \longleftarrow \mathcal{S}_n$ to obtain the two constituting sets), and $\mathcal{Z}^* := \{\boldsymbol{z}^*(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{X}\}$ the corresponding fixpoints. We now define the abstract semantics for a single step of the fixpoint solver as any sound abstract transformer $\boldsymbol{g}^\#_\alpha : \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A}$ of the iterated function $\boldsymbol{g}_\alpha : \mathbb{R}^{p+q} \mapsto \mathbb{R}^p$, i.e., any $\boldsymbol{g}^\#_\alpha$ satisfying $\gamma(\boldsymbol{g}^\#_\alpha(\hat{\mathcal{X}}, \hat{\mathcal{S}}_n)) \supseteq \{\boldsymbol{g}_\alpha(\boldsymbol{x}, \boldsymbol{s}) \mid \boldsymbol{x} \in \gamma(\hat{\mathcal{X}}), \boldsymbol{s} \in \gamma(\hat{\mathcal{S}}_n)\}$, for the abstract elements $\hat{\mathcal{X}}, \hat{\mathcal{S}}_n, \hat{\mathcal{Z}}_n, \hat{\mathcal{U}}_n$ from domain $\mathcal{A}$, which over-approximate the corresponding sets, e.g., $\gamma(\hat{\mathcal{X}}) \supseteq \mathcal{X}$.

We define the abstract semantics of the fixpoint solver Ξ yielding the concrete $\mathcal{Z}^*$ such that it satisfies $\gamma(\Xi^\#(\hat{\mathcal{X}})) \supseteq \{\Xi(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{X}\} =: \mathcal{Z}^*$. To this end, we derive the following theorem:

Theorem 3.1 (Fixpoint contraction). *Let*

- *$\boldsymbol{g}_\alpha$ be an iterative process, guaranteed to converge to a unique fixpoint $\boldsymbol{z}^*$ in finitely many steps for any bounded input $\boldsymbol{x}$, and $\boldsymbol{g}^\#_\alpha$ its sound abstract transformer,*
- *$\hat{\mathcal{S}}_{n+1} := \boldsymbol{g}^\#_\alpha(\hat{\mathcal{X}}, \hat{\mathcal{S}}_n)$ an abstract element in $\mathcal{A}$ describing a closed set and denoting an over-approximation of applying $\boldsymbol{g}_\alpha$ $(n+1)$-times for some $\boldsymbol{z}_0, \boldsymbol{u}_0$ on all inputs $\boldsymbol{x}' \in \hat{\mathcal{X}}$.*

Then for $[\hat{\mathcal{Z}}_n, \hat{\mathcal{U}}_n] \longleftarrow \hat{\mathcal{S}}_n$:

$$\hat{\mathcal{S}}_{n+1} \sqsubseteq \hat{\mathcal{S}}_n \quad \Longrightarrow \quad \mathcal{Z}^* \subseteq \gamma(\hat{\mathcal{Z}}_{n+1}). \tag{2}$$

Proof. We have

$$\mathcal{S}_{n+1} \subseteq \gamma(\hat{\mathcal{S}}_{n+1}) \subseteq \gamma(\hat{\mathcal{S}}_n) \tag{3}$$

where the first $\subseteq$ holds by definition and the second follows from the left-hand side of Eq. (2) and the definition of $\sqsubseteq$. Over-approximating $\mathcal{S}_{n+1}$ with $\hat{\mathcal{S}}_n \sqsupseteq \hat{\mathcal{S}}_{n+1}$, $\mathcal{S}_{n+2} \subseteq \gamma(\hat{\mathcal{S}}_{n+1})$ follows immediately via Eq. (3). Thus, $\mathcal{S}_j \subseteq \hat{\mathcal{S}}_{n+1}$ for $j > n$ follows by induction and thereby $\mathcal{Z}_j \subseteq \gamma(\hat{\mathcal{Z}}_{n+1}) \forall j > n$.

By the convergence guarantee of the concrete iteration, we have: for any $\epsilon \in \mathbb{R}^{>0}$ there exists a $l \in \mathbb{N}$ with $l \geq n+1$ such that we have $\|z_k - z^*\| \leq \epsilon, \forall k \in \mathbb{N}, k \geq l$. By the definition of $\mathcal{Z}_k$ we also have $z_k \in \mathcal{Z}_k$. For $\epsilon \longrightarrow 0$ and hence $\|z_k - z^*\| \longrightarrow 0$ it follows that $z^* \in \mathcal{Z}_k \cup \partial\mathcal{Z}_k =: \overline{\mathcal{Z}}_k$. Thus for each $\boldsymbol{x} \in \mathcal{X}$, there exists a $k_{\boldsymbol{x}}$ such that $z^*(\boldsymbol{x}) \in \overline{\mathcal{Z}}_{k_{\boldsymbol{x}}} \subseteq \gamma(\hat{\mathcal{Z}}_{n+1})$ due to the closedness of $\gamma(\hat{\mathcal{Z}})$. Finally, $\mathcal{Z}^* \subseteq \bigcup_{x \in \mathcal{X}} \overline{\mathcal{Z}}_{k_x} \subseteq \gamma(\hat{\mathcal{Z}}_{n+1})$. □

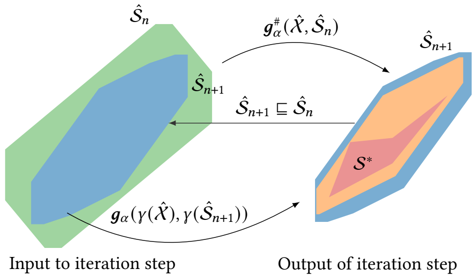

$\hat{\mathcal{S}}_n$
$g^\#_\alpha(\hat{\mathcal{X}}, \hat{\mathcal{S}}_n)$
$\hat{\mathcal{S}}_{n+1}$
$\hat{\mathcal{S}}_{n+1}$
$\hat{\mathcal{S}}_{n+1} \sqsubseteq \hat{\mathcal{S}}_n$
$\mathcal{S}^*$
$g_\alpha(\gamma(\hat{\mathcal{X}}), \gamma(\hat{\mathcal{S}}_{n+1}))$
Input to iteration step
Output of iteration step

Fig. 6. If an over-approximated solver iteration $\hat{\mathcal{S}}_{n+1} = g^\#_\alpha(\hat{\mathcal{X}}, \hat{\mathcal{S}}_n)$ (blue) is contained in the previous state $\hat{\mathcal{S}}_n$ (green), any (exact) iteration of $g_\alpha(\gamma(\hat{\mathcal{X}}), \gamma(\hat{\mathcal{S}}_{n+1}))$ (orange) will not escape from $\hat{\mathcal{S}}_{n+1}$. This implies containment of the true fixpoint set (red) $\mathcal{S}^* \subseteq \gamma(\hat{\mathcal{S}}_{n+1})$.

Intuitively, if we consistently apply $g^\#_\alpha$ until we detect contraction ($\hat{\mathcal{S}}_{n+1} \sqsubseteq \hat{\mathcal{S}}_n$), then $\hat{\mathcal{S}}_n$ is a so-called post-fixpoint and its concretization includes the true fixpoint set $\mathcal{Z}^* \subseteq \gamma(\hat{\mathcal{Z}}_{n+1}) \subseteq \gamma(\hat{\mathcal{Z}}_n)$. Below, we provide an intuition along the illustration in Fig. 6: Once an iteration of $g^\#_\alpha$ maps $\hat{\mathcal{S}}_n$ (green in Fig. 6) to a subset of itself $\hat{\mathcal{S}}_{n+1} \sqsubseteq \hat{\mathcal{S}}_n$ (blue), it follows from the soundness of $g^\#_\alpha$ that any number $k > 0$ of further applications of the concrete g_α to $\gamma(\hat{\mathcal{S}}_{n+1}) \subseteq \gamma(\hat{\mathcal{S}}_n)$ will map into $\mathcal{S}_{n+1} \subseteq \gamma(\hat{\mathcal{S}}_{n+1})$ (orange) and thus never 'escape' $\gamma(\hat{\mathcal{S}}_{n+1})$. As g_α is guaranteed to converge in finitely many steps in the concrete, the previously obtained $\gamma(\hat{\mathcal{S}}_{n+1}) \supseteq \mathcal{S}_{n+1} \supseteq \mathcal{S}_{n+k}$ must thus contain the true fixpoint set $[\mathcal{Z}^*, \mathcal{U}^*] \leftarrow \mathcal{S}^*$ (red). Note that this does not necessarily hold for applications of the abstract transformer $g^\#_\alpha$, as it, in contrast to g_α, is not necessarily monotonic, i.e., $\hat{\mathcal{S}}_A \sqsubseteq \hat{\mathcal{S}}_B \not\Longrightarrow g^\#_\alpha(\hat{\mathcal{S}}_A) \sqsubseteq g^\#_\alpha(\hat{\mathcal{S}}_B)$.

Empirically, the abstractions found via Theorem 3.1 are often relatively loose. However, we can obtain a more precise abstraction by applying additional iterations of a fixpoint set preserving abstract solver $g^\#_\alpha$ to $\hat{\mathcal{S}}_{n+1}$.

Fixpoint Set Preservation. We call an abstract transformer $g^\#_\alpha$ of g_α fixpoint set preserving if and only if applying it to any abstract state $\hat{\mathcal{S}}_n$ that contains the true fixpoint set, i.e., $\gamma(\hat{\mathcal{S}}_n) \supseteq \mathcal{S}^*$, results in an abstract state $\hat{\mathcal{S}}_{n+1}$ that still contains the true fixpoint set $\mathcal{S}^*$. Formally:

Definition 3.2 (Fixpoint set preservation). We call an abstract transformer $g^\#_\alpha$ fixpoint set preserving if and only if $\mathcal{S}^* \subseteq \gamma(\hat{\mathcal{S}}_n) \implies \mathcal{S}^* \subseteq \gamma(g^\#_\alpha(\hat{\mathcal{X}}, \hat{\mathcal{S}}_n))$.

Indeed, we can show that a broad class of $g^\#_\alpha$ are fixpoint set preserving:

Theorem 3.3 (Fixpoint set preservation). *Every sound abstract transformer $g^\#_\alpha$ of a locally Lipschitz g_α with convergence guarantees in the concrete is fixpoint set preserving.*

Proof. To prove by contradiction, let $\tilde{z}$ be a point close to the fixpoint z^* s.t. $\|\tilde{z} - z^*\| \leq \epsilon$ with map $\tilde{z}' = g_\alpha(\boldsymbol{x}, \tilde{z}, \boldsymbol{u})$ under the fixpoint iterator. Let us further assume that $\forall \boldsymbol{u} \in \mathcal{U}^*$ for some $\mathcal{U}^*$ an application of $z' = g_\alpha(\boldsymbol{x}, z^*, \boldsymbol{u})$ does not map back to z^*, i.e., $\|z^* - z'\| > d$.

- Recall that $g_\alpha(\boldsymbol{x}, \tilde{z}, \boldsymbol{u})$ is locally Lipschitz with $L < \infty$ by assumption.
- It follows from $\|\tilde{z} - z^*\| \leq \epsilon$ that $\|g_\alpha(\boldsymbol{x}, \tilde{z}, \boldsymbol{u}) - g_\alpha(\boldsymbol{x}, z^*, \boldsymbol{u})\| = \|\tilde{z}' - z'\| \leq L\epsilon$.
- Hence by the inverse triangle inequality $\|z^* - \tilde{z}'\| \geq \big| \|z^* - z'\| - \|\tilde{z}' - z'\| \big| \geq d - L\epsilon$
- Choose $\epsilon < d/(L+1) \implies \|z^* - \tilde{z}'\| \geq d - L\epsilon > \epsilon$. Note that d does not depend on ϵ.
- It follows that $\|g_\alpha(\boldsymbol{x}, \tilde{z}, \boldsymbol{u}) - z^*\| > \epsilon \quad \forall \tilde{z} \in \{z \mid \|z - z^*\| \leq \epsilon\}$ which contradicts the convergence guarantee.

By contradiction, it follows that $\exists \boldsymbol{u} \in \mathcal{U}^* : z^* = g_\alpha(\boldsymbol{x}, z^*, \boldsymbol{u})$. □

Abstract Interpreter. The above results can be used to construct abstract interpreters for arbitrary locally Lipschitz iterative processes converging to unique fixpoints in finitely many steps. While we focus on the verification of monDEQs, we illustrate the wider applicability of our approach on a toy example of a square-root computation using the Householder method in Section 6.5.

To actually construct an abstract interpreter for high-dimensional problems based on the above results, we need a suitable abstract domain $\mathcal{A}$ equipped with an efficient containment check $\sqsubseteq$ and precise transformers $\boldsymbol{g}^{\#}_{\alpha}$ for the used fixpoint solver.

4 THE CH-ZONOTOPE ABSTRACT DOMAIN

In this section, we introduce the **C**ontaining-**H**ybrid-Zonotope (CH-Zonotope), a novel abstract domain that enables our efficient domain-specific abstract interpreter. Based on Zonotope [Ghorbal et al. 2009], our domain is designed to carefully balance three features: (i) efficient propagation of abstract elements, (ii) fast (abstract) inclusion checks, and (iii) precision of all abstract transformers needed for (i) and (ii). Recall that none of the abstract domains typically used for neural network verification satisfies all three of these requirements, as they were designed for neural architectures with a constant (small) number of layers (see Section 2.3).

Zonotope. We begin with a brief recap of the Zonotope domain [Ghorbal et al. 2009; Singh et al. 2018]. A Zonotope $\hat{\mathcal{Z}} \in \mathcal{A}$ describing a volume $\gamma(\hat{\mathcal{Z}}) \subseteq \mathbb{R}^p$, is defined as $\hat{\mathcal{Z}} = \boldsymbol{A}\boldsymbol{\nu} + \boldsymbol{a}$, where $\boldsymbol{A} \in \mathbb{R}^{p \times k}$ is called the error coefficient matrix, $\boldsymbol{a} \in \mathbb{R}^p$ the center, and $\boldsymbol{\nu} = [-1, 1]^k$ the Zonotope error terms. Its concretization function is defined as $\gamma(\hat{\mathcal{Z}}) \coloneqq \{\boldsymbol{x} = \boldsymbol{A}\boldsymbol{\nu} + \boldsymbol{a} \mid \boldsymbol{\nu} \in [-1, 1]^k\}$. Using the exact abstract transformer $f^{\#}(\hat{\mathcal{Z}})$ of Singh et al. [2018] for an affine transformations $f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{c}$, we obtain $\hat{\mathcal{Z}}'$ with $\boldsymbol{A}' = \boldsymbol{W}\boldsymbol{A}$ and $\boldsymbol{a}' = \boldsymbol{W}\boldsymbol{a} + \boldsymbol{c}$.

CH-Zonotope. We define a CH-Zonotope $\hat{\mathcal{Z}} \in \mathcal{A}$ describing a volume $\gamma(\hat{\mathcal{Z}}) \subseteq \mathbb{R}^p$ as

$$\hat{\mathcal{Z}} = \boldsymbol{A}\boldsymbol{\nu} + \operatorname{diag}(\boldsymbol{b})\boldsymbol{\eta} + \boldsymbol{a}, \tag{4}$$

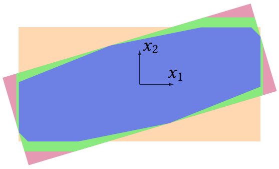


Fig. 7. Over-approximations of an improper CH-Zonotope (blue) by a proper one with (green) and without (red) Box component and a Box (orange).

by extending the Zonotope domain with the Box error vector $\boldsymbol{b} \in \left(\mathbb{R}^{\geq 0}\right)^p$ and corresponding Box error terms $\boldsymbol{\eta} = [-1, 1]^p$. If $\boldsymbol{A}$ is invertible, i.e. full rank and $k = p$, we call $\hat{\mathcal{Z}}$ a *proper* CH-Zonotope, or else an *improper* one. We adapt the concretization function to $\gamma(\hat{\mathcal{Z}}) = \{\boldsymbol{x} = \boldsymbol{A}\boldsymbol{\nu} + \operatorname{diag}(\boldsymbol{b})\boldsymbol{\eta} + \boldsymbol{a} \mid \boldsymbol{\nu} \in [-1, 1]^k, \boldsymbol{\eta} \in [-1, 1]^p\}$ and define a partial order $\sqsubseteq$ over CH-Zonotope based on the set inclusion $\subseteq$ of their concretizations. Formally, any CH-Zonotope can be seen as the Minkowski sum of a Zonotope ($\boldsymbol{A}\boldsymbol{\nu}$) and a Hyperbox ($\operatorname{diag}(\boldsymbol{b})\boldsymbol{\eta}$), also called Hybrid Zonotope [Goubault and Putot 2008; Mirman et al. 2018]. However, not every Hybrid-Zonotope is a *proper* CH-Zonotope as their error matrix is generally not invertible, which is crucial for our efficient containment check (discussed later).

Computing $\hat{\mathcal{Z}}' \sqsubseteq \hat{\mathcal{Z}}$ exactly is generally intractable. Therefore, we introduce an efficient over-approximation (discussed later) that is sound but not complete and requires the outer CH-Zonotope to be proper. By slight abuse of notation, we also denote it by $\sqsubseteq$. While a similar inclusion check is possible for any standard Zonotope with p linearly independent error terms ($\boldsymbol{b} = \boldsymbol{0}$), equivalent to a Parallelotope [Amato and Scozzari 2012], and any Box approximation ($\boldsymbol{A} = \boldsymbol{0}$), a CH-Zonotope yields a tighter abstraction than either since it can effectively employ twice as many error terms. We visualize this in Fig. 7, where we show a Box (orange), Parallelotope (red), and proper CH-Zonotope (green) abstraction of the original set (blue).

Abstract Transformers. For affine transformations, we use the Zonotope transformer described above, casting the Box errors as Zonotope errors by setting $\hat{\boldsymbol{A}} = [\boldsymbol{A}, \operatorname{diag}(\boldsymbol{b})]$ and $\hat{\boldsymbol{b}} = \boldsymbol{0}$ before applying the transformer. This yields an improper CH-Zonotope with a zero Box component. To encode the ReLU function, $y = \max(x, 0)$, for a CH-Zonotope $\hat{\mathcal{Z}}$, we modify the Zonotope transformer proposed by Singh et al. [2018] (recovered for $\boldsymbol{b} = \boldsymbol{0}$):

$$\hat{\mathcal{Z}}' = \boldsymbol{A}'\boldsymbol{\nu} + \operatorname{diag}(\boldsymbol{b}')\boldsymbol{\eta} + \boldsymbol{a}' = ReLU^{\#}_{\boldsymbol{\lambda}}(\hat{\mathcal{Z}})$$
$$\boldsymbol{A}' = \boldsymbol{\lambda}\boldsymbol{A}' \qquad \boldsymbol{b}' = \boldsymbol{\lambda}\boldsymbol{b} + \boldsymbol{\mu}$$
$$\boldsymbol{a}' = \boldsymbol{a} + \boldsymbol{\mu} \qquad \boldsymbol{\mu} = \begin{cases} (1-\boldsymbol{\lambda})\,\boldsymbol{u}_x/2 & \text{if } 0 \leq \boldsymbol{\lambda} \leq \boldsymbol{u}_x/(\boldsymbol{u}_x - \boldsymbol{l}_x) \\ -\boldsymbol{\lambda}\,\boldsymbol{l}_x/2 & \text{if } \boldsymbol{u}_x/(\boldsymbol{u}_x - \boldsymbol{l}_x) \leq \boldsymbol{\lambda} \leq 1 \end{cases}.$$

Applying this transformer will result in a CH-Zonotope with a non-zero Box component without changing its properness. By default, we choose $\lambda = u_x/(u_x - l_x)$ leading to the smallest volume in the 2d input-output space.

Consolidating Error Terms. To enable efficient inclusion checks and limit the number of error terms, we regularly over-approximate an improper CH-Zonotope $\hat{\mathcal{Z}}$ ($\boldsymbol{A} \in \mathbb{R}^{p \times k}$) with a proper one $\hat{\mathcal{Z}}'$ (invertible $\boldsymbol{A}' \in \mathbb{R}^{p \times p}$). We call this process *error consolidation.* If the Box component is zero ($\boldsymbol{b} = 0$) and $k > p$, this is known in the literature as order reduction via outer-approximation [Kopetzki et al. 2017; Sadraddini and Tedrake 2019]. If $k > p$, we consolidate the k old error terms into p new ones, thus reducing the representation size. We ensure that $\boldsymbol{A}'$ has full rank and is therefore invertible. If $k \leq p$, we pick a subset with full rank and complete it to a basis. In monDEQ certification, $p = \dim(\boldsymbol{z})$ is the size of the latent dimension.

Theorem 4.1 (Consolidating errors). *Let $\hat{\mathcal{Z}} = \boldsymbol{A}\boldsymbol{\nu} + \operatorname{diag}(\boldsymbol{b})\boldsymbol{\eta} + \boldsymbol{a}$ be an improper CH-Zonotope with $\boldsymbol{A} \in \mathbb{R}^{p \times k}$. Further, let $\tilde{\boldsymbol{A}} \in \mathbb{R}^{p \times p}$ be invertible. Then the proper CH-Zonotope $\hat{\mathcal{Z}}' = \boldsymbol{A}'\boldsymbol{e}'_1 + \operatorname{diag}(\boldsymbol{b})\boldsymbol{\eta} + \boldsymbol{a}$ with*

$$\boldsymbol{A}' = \operatorname{diag}(\boldsymbol{c})\tilde{\boldsymbol{A}} \qquad \text{where } \boldsymbol{c} = |\tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}|\boldsymbol{1} \tag{5}$$

is a sound over-approximation, i.e., $\hat{\mathcal{Z}}' \sqsupseteq \hat{\mathcal{Z}}$ of the improper one, where $\boldsymbol{1}$ denotes the k-dimensional one vector and $|\cdot|$ the elementwise absolute. We call $\boldsymbol{c}$ the consolidation coefficients.

Proof. Let w.l.o.g. $\boldsymbol{a} = \boldsymbol{0}$, $\boldsymbol{b} = \boldsymbol{0}$, and $\boldsymbol{A} \in \mathbb{R}^{p \times k}$. As $\tilde{\boldsymbol{A}}$ is a basis of $\mathbb{R}^p$ and hence invertible, we can write the contribution of every error term as $\boldsymbol{A}_j \nu_j = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{\nu}}'^{(j)}$ with $\tilde{\boldsymbol{\nu}}'^{(j)} = \tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}_j \nu_j$. As $\nu_j \in [-1, 1]$, we have $\tilde{\boldsymbol{\nu}}'^{(j)} \in \operatorname{diag}(\tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}_j)\tilde{\boldsymbol{\nu}}^{(j)}$ with $\tilde{\boldsymbol{\nu}}^{(j)} \in [-1, 1]^p$. This allows us to rewrite

$$\hat{\mathcal{Z}} = \begin{Bmatrix} \tilde{\boldsymbol{A}} \sum_{j=1}^{k} \tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}_j \nu_j \\ \forall\, \boldsymbol{\nu} \in [-1, 1]^k \end{Bmatrix} \subseteq \begin{Bmatrix} \tilde{\boldsymbol{A}} \sum_{j=1}^{k} \operatorname{diag}(\tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}_j)\tilde{\boldsymbol{\nu}}^{(j)} \\ \forall\, \tilde{\boldsymbol{\nu}}^{(1)}, \dots, \tilde{\boldsymbol{\nu}}^{(k)} \in [-1, 1]^p \end{Bmatrix} = \begin{Bmatrix} \tilde{\boldsymbol{A}} \operatorname{diag}(|\tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}|\boldsymbol{1})\tilde{\boldsymbol{\nu}} \\ \forall\, \tilde{\boldsymbol{\nu}} \in [-1, 1]^p \end{Bmatrix} = \hat{\mathcal{Z}}',$$

where the second last equality follows from linearity and the choice $\tilde{\boldsymbol{\nu}}_j = \pm \operatorname{sign}(\tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}_j)$. □

The intuition behind this approximation is shown in Fig. 8, where we over-approximate the green $\hat{\mathcal{Z}}$ with the gray $\hat{\mathcal{Z}}'$ (choosing a suboptimal basis for illustration purposes). All error vectors (columns) in the old error matrix $\boldsymbol{A}$ (shown as solid red and blue arrows) are first decomposed into a linear combination of error vectors in the new basis $\tilde{\boldsymbol{A}}$ (dashed red and blue arrows). Then, the absolute values (to correct for their orientation) of these contributions $|\tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}|$ are summed up over all

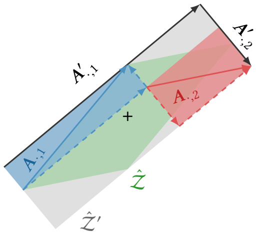
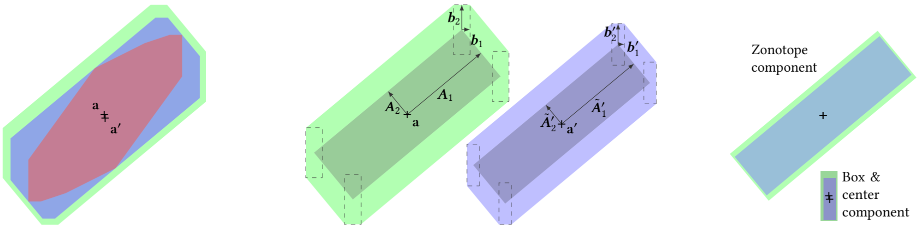


Fig. 8. Illustration of error consolidation via Theorem 4.1. All vectors are scaled by factor 2.



(a) True inclusion

(b) Decomposition of CH-Zonotopes into their Box and Zonotope components

(c) Inclusion check for both components

Fig. 9. Illustration of checking the containment of an improper CH-Zonotope (red) in a proper CH-Zonotope (green), by consolidating errors (blue). Fig. 9b shows the proper CH-Zonotopes decomposed into their Box and Zonotope components. In Fig. 9c we illustrate the containment check of these components individually.

error vectors to obtain the consolidation coefficients $\boldsymbol{c}$. Finally, we multiply these consolidation coefficients with the error directions of the new basis $\tilde{\boldsymbol{A}}$ to obtain the new error matrix $\boldsymbol{A}'_{\cdot,i} = c_i \tilde{\boldsymbol{A}}_{\cdot,i}$ (solid black arrows). We note that this has complexity $O(p^2(p+k))$.

Choosing the New Error Basis. To minimize the imprecision incurred when consolidating error terms, a suitable basis $\tilde{\boldsymbol{A}}$ has to be chosen. We use the PCA-basis of the original error matrix $\boldsymbol{A}$, as it has been empirically shown to yield the tightest approximation while being computationally feasible in high dimensions [Kopetzki et al. 2017].

Inclusion Checks for CH-Zonotope. Enabling efficient inclusion checks in high dimensions is one of the main motivations for the CH-Zonotope domain. Here, we first provide a high-level outline of our approach, illustrated in Fig. 9, before giving more detail on the individual steps.

We aim to determine whether the proper CH-Zonotope $\hat{\mathcal{Z}} = \boldsymbol{A}\boldsymbol{\nu} + \text{diag}(\boldsymbol{b})\boldsymbol{\eta} + \boldsymbol{a}$ (green in Fig. 9a) contains the improper $\hat{\mathcal{Z}}' = \boldsymbol{A}'\boldsymbol{\nu} + \text{diag}(\boldsymbol{b}')\boldsymbol{\eta} + \boldsymbol{a}'$ (red), i.e., $\hat{\mathcal{Z}} \sqsupseteq \hat{\mathcal{Z}}'$. At a high level, we first consolidate the errors of the improper $\hat{\mathcal{Z}}'$ (blue) before decomposing both CH-Zonotope into their Zonotope and Box components (shown in Fig. 9b) and checking whether the outer components contain their respective inner counterparts (shown in Fig. 9c).

To determine containment of the Zonotope component, we consolidate the error matrix $\boldsymbol{A}'$ with basis $\boldsymbol{A}$ as discussed above (shown in blue in Fig. 9a). This leads to perfectly aligned error vectors, enabling us to directly compare their lengths. If all error terms of the consolidated $\tilde{\boldsymbol{A}}'$ are shorter than their counterparts in $\boldsymbol{A}$, the Zonotope components are contained (shown overlayed in Fig. 9c). More efficiently, we only compute the consolidation coefficients and check $\boldsymbol{c} = |\boldsymbol{A}^{-1}\boldsymbol{A}'|\boldsymbol{1} < \boldsymbol{1}$.

To show containment of the Box components, we can simply check that $\boldsymbol{b}' \leq \boldsymbol{b}$. However, we observe that negative values in the difference vector $\boldsymbol{b}' - \boldsymbol{b}$ denote directions in which $\boldsymbol{b}$ is larger than $\boldsymbol{b}'$ and can hence compensate for differences in the center terms $\boldsymbol{a}' - \boldsymbol{a}$. Positive values in $\boldsymbol{b}' - \boldsymbol{b}$ denote directions in which $\boldsymbol{b}$ is too small to cover $\boldsymbol{b}'$. Combining these two, we obtain a residual Box component $\boldsymbol{d} = \max(0, |\boldsymbol{a}' - \boldsymbol{a}| + \boldsymbol{b}' - \boldsymbol{b})$ that needs to additionally be covered by the Zonotope component. To this end, we can cast $\boldsymbol{d}$ as additional error terms of $\boldsymbol{A}'$ and update the Zonotope inclusion check to $|\boldsymbol{A}^{-1}\boldsymbol{A}'|\boldsymbol{1} + |\boldsymbol{A}^{-1}\,\text{diag}(\boldsymbol{d})|\boldsymbol{1} < \boldsymbol{1}$. This compensation is not necessary in Fig. 9. We formalize this containment check as follows:

Theorem 4.2 (CH-Zonotope Containment). *Let $\hat{\mathcal{Z}} = \boldsymbol{A}\boldsymbol{\nu} + \text{diag}(\boldsymbol{b})\boldsymbol{\eta} + \boldsymbol{a}$ be a proper CH-Zonotope and $\hat{\mathcal{Z}}' = A'\boldsymbol{\nu}' + \text{diag}(\boldsymbol{b}')\boldsymbol{\eta}' + \boldsymbol{a}'$ an improper one. $\hat{\mathcal{Z}}'$ is contained in $\hat{\mathcal{Z}}$ if*

$$\left|\boldsymbol{A}^{-1}\boldsymbol{A}'\right|\boldsymbol{1} + \left|\boldsymbol{A}^{-1}\,\text{diag}\left(\max\left(\boldsymbol{0}, |\boldsymbol{a}' - \boldsymbol{a}| + \boldsymbol{b}' - \boldsymbol{b}\right)\right)\right|\boldsymbol{1} \leq \boldsymbol{1} \tag{6}$$

holds element-wise. Where $\boldsymbol{A}^{-1}$ always exists as $\hat{\mathcal{Z}}$ is proper and therefore $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ invertible.

Proof. Containment is equivalent to showing that for all points in $\hat{\mathcal{Z}}'$, described by error terms $\boldsymbol{\nu}' \in [-1,1]^k, \boldsymbol{\eta}' \in [-1,1]^p$, there exist $\boldsymbol{\nu} \in [-1,1]^p, \boldsymbol{\eta} \in [-1,1]^p$ of $\hat{\mathcal{Z}}$ such that:

$$\boldsymbol{A}\boldsymbol{\nu} + \mathrm{diag}(\boldsymbol{b})\boldsymbol{\eta} + \boldsymbol{a} = \boldsymbol{A}'\boldsymbol{\nu}' + \mathrm{diag}(\boldsymbol{b}')\boldsymbol{\eta}' + \boldsymbol{a}'.$$

We subtract $\boldsymbol{a}$ from both sides and over-approximate the right-hand side by increasing the Box size by the absolute center difference $|\boldsymbol{a}' - \boldsymbol{a}|$ yielding $\boldsymbol{b}'' := \boldsymbol{b}' + |\boldsymbol{a}' - \boldsymbol{a}|$. We now show that we can chose $\boldsymbol{\nu}, \boldsymbol{\eta}$ such that $\boldsymbol{A}\boldsymbol{\nu} + \mathrm{diag}(\boldsymbol{b})\boldsymbol{\eta} = \boldsymbol{A}'\boldsymbol{\nu}' + \mathrm{diag}(\boldsymbol{b}'')\boldsymbol{\eta}''$ holds. We choose

$$\boldsymbol{\eta} = \mathrm{diag}(1/\boldsymbol{b})\,\mathrm{sign}(\boldsymbol{\eta}'')\min(\boldsymbol{b}, \mathrm{diag}(\boldsymbol{b}'')|\boldsymbol{\eta}''|),$$

guaranteed to yield $|\boldsymbol{\eta}| \leq 1$ and obtain by substitution

$$\begin{aligned}
\boldsymbol{A}\boldsymbol{\nu} &= \boldsymbol{A}'\boldsymbol{\nu}' + \max(0, \mathrm{diag}(\boldsymbol{b}'')\boldsymbol{\eta}'' - \boldsymbol{b}) \\
\boldsymbol{\nu} &= \boldsymbol{A}^{-1}\boldsymbol{A}'\boldsymbol{\nu}' + \boldsymbol{A}^{-1}\max(\boldsymbol{0}, \mathrm{diag}(\boldsymbol{b}'')\boldsymbol{\eta}'' - \boldsymbol{b}) \\
&\overset{(*)}{\leq} |\boldsymbol{A}^{-1}\boldsymbol{A}'|\boldsymbol{1} + |\boldsymbol{A}^{-1}\,\mathrm{diag}(\max(\boldsymbol{0}, \boldsymbol{b}'' - \boldsymbol{b}))|\boldsymbol{1} \overset{(**)}{\leq} \boldsymbol{1},
\end{aligned}$$

where in $(*)$ we use the relation shown in Theorem 4.1 for the sound representation of a decomposition and the fact that setting $\boldsymbol{\eta}''$ to a 1-vector maximizes $\mathrm{diag}(\boldsymbol{b}'')\boldsymbol{\eta}''$ and $(**)$ follows directly from the condition of Eq. (6). Taking the absolute value we obtain $|\boldsymbol{\nu}| \leq \boldsymbol{1}$. □

In contrast to exact containment checks for general Zonotope which are co-NP-complete and hence infeasible Kulmburg and Althoff [2021], our Theorem 4.2 constitutes a sound but not complete check with complexity $\mathcal{O}(p^2(p+k))$. Another approximate method with polynomial time complexity was proposed by Sadraddini and Tedrake [2019], which they show to be close to loss-less in low dimensions ($p \leq 10$). However, their method involves solving a linear program in $\mathcal{O}(k_{\text{inner}}k_{\text{outer}})$ variables with $\mathcal{O}(pk_{\text{inner}})$ constraints, where $k_{\text{inner}} \geq p$ and $k_{\text{outer}} \geq p$ are the number of error terms and p is the dimensionality. Making generous assumptions on the number of error terms and the complexity of the LP-solver [Jiang et al. 2020], this leads to an overall complexity of $\tilde{\mathcal{O}}(p^6)$, which makes it practically intractable for our use-case, as we will show later (see Section 6.4).

5 APPLICATION TO FIXPOINT-BASED NEURAL NETWORKS

In this section, we first introduce (monotone Operator) Deep Equilibrium Models (monDEQs) for concrete points in Section 5.1 before considering their abstraction in Section 5.2.

5.1 Deep Equilibrium Models on Points

Deep Equilibrium Models (DEQs). Implicit-Layer [Amos and Kolter 2017; Ghaoui et al. 2021] and Deep Equilibrium Models (DEQs) [Bai et al. 2019] were recently introduced to enable more memory-efficient model parameterizations. Unlike traditional deep neural networks, which propagate inputs through a finite number of different layers, DEQs conceptually apply the same layer repeatedly until converged to a fixpoint, corresponding to an infinite depth model with parameter-sharing. A DEQ $\boldsymbol{h}$ obtains its final prediction $\boldsymbol{y}$ by applying a linear layer to this fixpoint:

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) := \boldsymbol{V}\boldsymbol{z}^* + \boldsymbol{v}, \qquad \boldsymbol{z}^* = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}^*). \tag{7}$$

Monotone Operator Deep Equilibrium Models (monDEQs). A major drawback of general DEQs is that neither the existence nor uniqueness of their fixpoints is guaranteed. To address this issue, Winston and Kolter [2020] introduced monDEQs as a particular form of DEQs guaranteed to have a unique fixpoint by parametrizing

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}) = \sigma(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}) \tag{8}$$

with $\boldsymbol{x} \in \mathbb{R}^q, \boldsymbol{z} \in \mathbb{R}^p, \boldsymbol{U} \in \mathbb{R}^{p\times q}, \boldsymbol{W} = (1-m)\boldsymbol{I} - \boldsymbol{P}^T\boldsymbol{P} + \boldsymbol{Q} - \boldsymbol{Q}^T$ where $\boldsymbol{P}, \boldsymbol{Q} \in R^{p\times p}$, and monotonicity parameter $m > 0$. These existence and uniqueness properties allow a certification of monDEQs that is independent of how a fixpoint was obtained, yielding far stronger guarantees than possible in the general DEQ setting. Throughout this paper, we will focus on the ReLU activation (i.e., $\sigma := ReLU$).

Fixpoint solvers. As discussed in Section 3, iteratively applying $\boldsymbol{f}$ often does not converge and iterative fixpoint solvers which converge to the unique fixpoint under mild conditions are employed instead. For a monDEQ $\boldsymbol{h}$ with iteration function $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}) = ReLU(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{U}\boldsymbol{x} + \boldsymbol{b})$, we let $\boldsymbol{g}$ denote an iteration of a fixpoint solver using operator splitting:

- **Forward-Backward Splitting (FB)** where $\boldsymbol{s}_{n+1}$ is computed as
$$\boldsymbol{s}_{n+1} := \boldsymbol{g}_\alpha^{FB}(\boldsymbol{x}, \boldsymbol{s}_n) = ReLU((1-\alpha)\boldsymbol{s}_n + \alpha(\boldsymbol{W}\boldsymbol{s}_n + \boldsymbol{U}\boldsymbol{x} + \boldsymbol{b})), \tag{9}$$
converging to $\boldsymbol{s}^* = \boldsymbol{z}^*$ of $\boldsymbol{f}$, for any $0 < \alpha < \frac{2m}{\|\boldsymbol{I}-\boldsymbol{W}\|_2^2}$ [Winston and Kolter 2020].
- **Peaceman-Rachford Splitting (PR)** where $\boldsymbol{s}_{n+1} := \boldsymbol{g}_\alpha^{PR}(\boldsymbol{x}, \boldsymbol{s}_n)$ is computed as
$$\begin{aligned} [\boldsymbol{z}_n; \boldsymbol{u}_n] &\leftarrow \boldsymbol{s}_n & \boldsymbol{u}_{n+1} &= 2\boldsymbol{z}_{n+1/2} - \boldsymbol{u}_{n+1/2} \\ \boldsymbol{u}_{n+1/2} &= 2\boldsymbol{z}_n - \boldsymbol{u}_n & \boldsymbol{z}_{n+1} &= ReLU(\boldsymbol{u}_{n+1}) \\ \boldsymbol{z}_{n+1/2} &= (\boldsymbol{I} + \alpha(\boldsymbol{I} - \boldsymbol{W}))^{-1}(\boldsymbol{u}_{n+1/2} + \alpha(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b})) & \boldsymbol{s}_{n+1} &\leftarrow [\boldsymbol{z}_{n+1}; \boldsymbol{u}_{n+1}]. \end{aligned} \tag{10}$$
PR splitting converges to $\boldsymbol{z}^*$ for any $\alpha > 0$ [Ryu and Boyd 2016].

While there exist many similar strategies, we restrict our discussion to the above examples. For both, we initialize $\boldsymbol{s}_0 = \boldsymbol{0}$ and write $\boldsymbol{g}_\alpha(\boldsymbol{x}, \boldsymbol{s}_n)$ for one iteration and $[\boldsymbol{z}; \boldsymbol{u}] \leftarrow \boldsymbol{s}$ for the unpacking of the latent state for both PR and FB. For the latter, we simply assume $\boldsymbol{u}_n$ to be zero-dimensional.

Both Forward-Backward (FB, Eq. (9)) and Peaceman-Rachford splitting (PR, Eq. (10)) are guaranteed to converge to the fixpoint of $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z})$, as defined in Eq. (7). In practice, they are iterated until $\|\boldsymbol{z}_n - \boldsymbol{z}_{n-1}\|$ becomes smaller than a predetermined stopping criterion, yielding $\boldsymbol{z}_n \approx \boldsymbol{z}^*(\boldsymbol{x})$.

Example (cont.) Our example from Eq. (1) is a monDEQ using FB splitting and parametrized with:
$$m = 4,\ \alpha = \tfrac{1}{10},\ P = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right),\ Q = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\right)\ W = \left(\begin{smallmatrix} -4 & -1 \\ 1 & -4 \end{smallmatrix}\right),\ U = \left(\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}\right),\ b = \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right).$$
For these parameters, the iterative functions are
$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}_n) &= ReLU\left(\left(\begin{smallmatrix} -4 & -1 \\ 1 & -4 \end{smallmatrix}\right)\boldsymbol{z}_n + \left(\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}\right)\boldsymbol{x} + \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)\right) \\ \boldsymbol{g}_\alpha(\boldsymbol{x}, \boldsymbol{s}_n) &= ReLU\left(\tfrac{1}{10}(\boldsymbol{W} + 9\boldsymbol{I})\boldsymbol{s}_n + \tfrac{1}{10}\left(\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}\right)\boldsymbol{x} + \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)\right) = ReLU\left(\tfrac{1}{10}\left(\begin{smallmatrix} 5 & 1 \\ -1 & 5 \end{smallmatrix}\right)\boldsymbol{s}_n + \tfrac{1}{10}\left(\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}\right)\boldsymbol{x}\right). \end{aligned}$$
Observe that $0 < \alpha = \frac{1}{10} < \frac{2m}{\|\boldsymbol{I}-\boldsymbol{W}\|_2^2} \approx 0.1538$. Interestingly, directly iterating $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}_n)$ diverges in our example, highlighting the importance of a suitable iterative solver.

5.2 Abstract Interpretation of monDEQs

Equipped with the building blocks discussed so far, we now introduce our abstract interpreter, Craft (**C**onvex **R**elaxation **A**bstract **F**ixpoint i**T**eration).

At a high level, given an input $\boldsymbol{x}$, a precondition φ, and a postcondition ψ over a monDEQ $\boldsymbol{h}$, Craft iteratively applies an abstract solver iteration $\boldsymbol{g}_\alpha^\#$ to CH-Zonotope abstractions of the input and solver state until our inclusion check (Theorem 4.2) can show that the resulting state $\hat{\mathcal{S}}_{n+1}$ is contained in the previous one $\hat{\mathcal{S}}_n$. By the contraction-based termination condition of Theorem 3.1, we have thus found an over-approximation $\hat{\mathcal{Z}}^*$ of the true fixpoint set $\mathcal{Z}^*$. Propagating the corresponding CH-Zonotope $\hat{\mathcal{Z}}^*$ through the last layer to obtain the CH-Zonotope abstraction of the output $\hat{\mathcal{Y}} = \boldsymbol{V}\hat{\mathcal{Z}}^* + \boldsymbol{v}$ (by slight abuse of notation), we check the postcondition $\psi(\hat{\mathcal{Y}})$. Below, we first discuss this process, outlined in Algorithm 1, informally, before formally showing its correctness.

While Craft is applicable to general pre- and postconditions φ and ψ, the presented version assumes that $\boldsymbol{x} \in \varphi(\boldsymbol{x})$ and that ψ is a statement over the outputs of the monDEQ $\boldsymbol{h}$. In particular, we focus on ℓ_∞ robustness certification, where $\varphi(\boldsymbol{x}) := \{\boldsymbol{x}' \mid \|\boldsymbol{x}-\boldsymbol{x}'\|_\infty \leq \epsilon\}$ and $\psi := \boldsymbol{h}_t(\boldsymbol{x}')-\boldsymbol{h}_i(\boldsymbol{x}') > 0, \forall i \neq t$. That is, the monDEQ $\boldsymbol{h}$ classifies all $\boldsymbol{x}'$ in an ℓ_∞-ball around $\boldsymbol{x}$ as class t.

What to Certify. As previously discussed and in agreement with prior work [Chen et al. 2021; Pabbaraju et al. 2021], Craft certifies properties for the true mathematical fixpoints $\boldsymbol{z}^*$, rather than any particular solver behavior. This yields stronger certificates as solvers are guaranteed to converge to these unique fixpoints with arbitrary precision.

Craft. Craft can be divided into two stages: First, it leverages the contraction-based termination condition of Theorem 3.1 to compute a first abstraction of the fixpoint set for a given precondition φ. Second, it tightens this abstraction by leveraging fixpoint set preservation (Definition 3.2) to show the postcondition ψ. This is detailed in Algorithm 1. We start by initializing, by slight abuse of notation, $\hat{\mathcal{Z}}_0 = \hat{\mathcal{U}}_0 = \{\boldsymbol{z}^*(\boldsymbol{x})\}$ to a concrete fixpoint (line 2). To compute an abstract fixpoint set via Theorem 3.1, we perform iterations of the $\neg$ `contained` branch (lines 6-8). In each iteration, we first consolidate the errors of the current abstraction (line 6) via Theorem 4.1, then perform one step of $\boldsymbol{g}^{\#,1}_{\alpha_1}$ (line 7), and finally check inclusion $\hat{\mathcal{S}}_{n+1} \sqsubseteq \hat{\mathcal{S}}_n$ via Theorem 4.2 (line 8). After detecting containment, we aim to tighten the thus obtained fixpoint abstraction in order to show ψ. To this end, we perform iterations of the `contained` branch (lines 10-14). Here, we apply $\boldsymbol{g}^{\#,2}_{\alpha_2}$ (line 10) and the classification layer (line 12) before checking ψ on the resulting CH-Zonotope (line 13).

Algorithm 1: Craft

```
Input: $x$, precondition $\varphi$, postcondition $\psi$, monDEQ $h$
Output: whether $\varphi(x) \models \psi(h(x))$
1  $\hat{\mathcal{X}} \leftarrow \varphi(x)$
2  $\hat{\mathcal{S}}_0 \leftarrow \{[z^*(x); z^*(x)]\}$        // $\hat{\mathcal{Z}}_0, \hat{\mathcal{U}}_0 \leftarrow \{z^*\}, \{z^*\}$
3  contained $\leftarrow$ false
4  for $n \leftarrow 1, \dots, n_{max}$ do
5      if $\neg$ contained
6          $\hat{\mathcal{S}}_n \leftarrow$ consolidate($\hat{\mathcal{S}}_n$)              // proper
7          $\hat{\mathcal{S}}_{n+1} \leftarrow g^{\#,1}_{\alpha_1}(\hat{\mathcal{X}}, \hat{\mathcal{S}}_n)$              // improper
8          contained $\leftarrow \hat{\mathcal{S}}_{n+1} \sqsubseteq \hat{\mathcal{S}}_n$
9      else
10         $\hat{\mathcal{S}}_{n+1} \leftarrow g^{\#,2}_{\alpha_2}(\hat{\mathcal{X}}, \hat{\mathcal{S}}_n)$              // improper
11         $[\hat{\mathcal{Z}}_{n+1}; \hat{\mathcal{U}}_{n+1}] \leftarrow \hat{\mathcal{S}}_{n+1}$
12         $\hat{\mathcal{Y}} \leftarrow V\hat{\mathcal{Z}}_{n+1} + v$                   // improper
13         if $\psi(\hat{\mathcal{Y}})$
14             return true
15 return false
```

Iterator Requirements. While Algorithm 1 does not require a specific operator splitting method $\boldsymbol{g}_\alpha$, $\boldsymbol{g}^{\#,1}_{\alpha_1}$ has to be chosen such that Theorem 3.1 on the contraction based termination condition is applicable and $\boldsymbol{g}^{\#,2}_{\alpha_2}$ has to be fixpoint-preserving according to Definition 3.2.

More concretely, we require the abstract transformer $\boldsymbol{g}^{\#,1}_{\alpha_1}$ (line 7) to be a sound abstraction of an operator splitting method $\boldsymbol{g}_{\alpha_1}$ which, in the concrete, is guaranteed to converge to a unique fixpoint in finitely many steps. And we require $\boldsymbol{g}^{\#,2}_{\alpha_2}$ (line 10) to be fixpoint set preserving (Definition 3.2), i.e., to map fixpoints upon themselves. By Theorem 3.3 the latter is the case for all $\boldsymbol{g}^{\#}_{\alpha_2}$ abstracting a fixed, locally-Lipschitz operator splitting method $\boldsymbol{g}_{\alpha_2}$ with convergence guarantees, including PR and FB. However, the following points have to be considered: PR iterations use auxiliary variables $\boldsymbol{u}$ which depend on α. Consequently, fixpoint set preservation is only guaranteed for a fixed α, preventing us from optimizing α to obtain tighter over-approximations. This limitation does not apply to FB splitting, as it does not use any auxiliary variables. However, the convergence requirement of Theorem 3.3 still limits α to $0 < \alpha < 2m/\|\boldsymbol{I}-\boldsymbol{W}\|_2^2$. We now lift this restriction, allowing us to apply further iterations of $\boldsymbol{g}^{FB}_{\alpha}$ with arbitrary $\alpha \in [0, 1]$ to tighten $\hat{\mathcal{S}}_n$ after showing containment with any method:

Theorem 5.1 (Fixpoint set preservation for FB splitting). *Every sound abstract transformer $\boldsymbol{g}_\alpha^{FB\#}$ of $\boldsymbol{g}_\alpha^{FB}$ is fixpoint set preserving for $0 \leq \alpha \leq 1$.*

Proof. For $\boldsymbol{g}_\alpha^{FB}$ we have $\boldsymbol{s} := \boldsymbol{z}$ and thus as per Eq. (9) for every concrete fixpoint $\boldsymbol{z}_n = \boldsymbol{z}^*$:

$$\boldsymbol{z}_{n+1} = ReLU((1-\alpha)\boldsymbol{z}^* + \alpha(\underbrace{\boldsymbol{W}\boldsymbol{z}^* + \boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}}_{\boldsymbol{z}'})) = \boldsymbol{z}^*$$

Element-wise, we have $z^* = 0$ if $z' \leq 0$ as $z^* = f(x, z^*) = ReLU(z')$ and else $z^* = z'$. Thus

$$ReLU((1-\alpha)z^* + \alpha z') = \begin{Bmatrix} ReLU((1-\alpha)0 \ + \alpha z') & \text{if } z' \leq 0 \\ ReLU((1-\alpha)z^* + \alpha z^*) & \text{else} \end{Bmatrix} = z^*.$$

As, in the concrete, every step of Forward-Backward splitting maps any fixpoint upon itself, any sound $\hat{\mathcal{Z}}_{n+1} = \boldsymbol{g}_\alpha^{FB\#}(\mathcal{X}, \hat{\mathcal{Z}}_n)$ will include all fixpoints contained in $\hat{\mathcal{Z}}_n$. □

Choice of Iterator. Given the constraints discussed above, we choose different algorithms for $\boldsymbol{g}_{\alpha_1}^{\#,1}$ and $\boldsymbol{g}_{\alpha_2}^{\#,2}$, optimizing for containment and tight final abstractions, respectively. For $\boldsymbol{g}_{\alpha_1}^{\#,1}$ we typically use PR, as it empirically is significantly less sensitive to hyperparameter choices (see Fig. 12) and contracts to the actual fixpoint set more quickly [Winston and Kolter 2020]. For $\boldsymbol{g}_{\alpha_2}^{\#,2}$, both PR $\boldsymbol{g}_\alpha^{PR\#}$ and FB $\boldsymbol{g}_\alpha^{FB\#}$ are used depending on the underlying problem. In some settings the stronger contractive properties of PR yield tighter abstractions while in others choosing an optimal dampening parameter α via line search for FB works better. We further discuss this in Section 6.3.

Expansion. The key to showing containment is not the absolute tightness of the abstract iteration state $\hat{\mathcal{S}}_n$, but rather how much it tightens under application of $\boldsymbol{g}_{\alpha_1}^{\#,1}$. As further tightening an already very tight approximation can be challenging, we – perhaps counter-intuitively – expand our over-approximation as part of the error consolidation by setting

$$c = (1 + w_{mul})|\tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}|\boldsymbol{1} + w_{add}\boldsymbol{1} \tag{11}$$

in Eq. (5) until containment is found. Here, $w_{mul}, w_{add} \geq 0$ are the multiplicative and additive expansion parameters, respectively. The resulting looseness between the current approximation and the exact fixpoint set can make tightening the approximation and hence showing containment easier. As this expansion leads to a strictly larger over-approximation, it is a sound operation.

While this is similar to widening [Cousot and Cousot 1992b] at first glance, it aims to break a non-monotonic iteration of incomparable abstractions instead of ensuring termination of an otherwise infinite iteration of monotonically increasing abstractions.

Correctness. We now show the correctness of Craft w.r.t. to the concrete and abstract semantics defined in Section 3, instantiated for monDEQs.

Theorem 5.2 (Soundness of Craft). *For sound $\boldsymbol{g}_{\alpha_1}^{\#,1}$ fulfilling Theorem 3.1 and sound $\boldsymbol{g}_{\alpha_2}^{\#,2}$ fulfilling fixpoint-preservation (Definition 3.2), Algorithm 1 is sound. In particular:*

(1) Once `contained` ($\hat{\mathcal{S}}_{n+1} \sqsubseteq \hat{\mathcal{S}}_n$), $\hat{\mathcal{S}}_{n+1}$ contains the true fixpoint set.
(2) Algorithm 1 returns `true` only if $\varphi(\boldsymbol{x}) \models \psi(\boldsymbol{h}(\boldsymbol{x}))$.

Proof. (1) follows directly from the soundness of the containment check (Theorem 4.2) and the contraction-based termination criterion of Theorem 3.1 and (2) from Theorems 3.3 and 5.1 and the use of sound abstract transformers. □

Completeness. While Craft is sound, it is not complete. In particular, `consolidate`, `expand`, $\boldsymbol{g}_{\alpha_1}^{\#,1}$, and $\boldsymbol{g}_{\alpha_2}^{\#,2}$ are sources of imprecision. The inclusion check is also sound, but not complete.

Table 2. Overview of the obtained natural accuracy (*Acc.*), adversarial accuracy (*Bound*), the number of samples for which we found a fixpoint over-approximation (*Cont.*), the certified accuracy (*Cert.*), and the average time per correctly classified sample for the first 100 samples from the corresponding test set.

Dataset	*Model*	*Latent Size*	*# Acc.*	ϵ	*# Bound*	*# Cont.*	*# Cert.*	*Time [s]*
MNIST	`FCx40`	40	99	0.05	70	100	36	17.2
	`FCx87`	87	99	0.05	75	100	30	15.8
	`FCx100`	100	96	0.05	73	100	24	13.2
	`FCx200`	200	99	0.05	80	100	26	14.0
	`ConvSmall`	648	97	0.05	80	100	68	22.4
CIFAR10	`FCx200`	200	63	2/255	36	100	22	16.8
	`ConvSmall`	800	55	2/255	32	100	29	41.1

Generality. Craft can be instantiated with any abstract domain supporting the required abstract transformers. Only the consolidation in line 6 is specific to CH-Zonotope and it can be removed without affecting Craft's soundness. However, while all domains discussed in Section 2 possess the required transformers, only CH-Zonotope combines sufficient precision with tractable containment checks and efficient propagation (see Section 6.4).

6 EXPERIMENTAL EVALUATION

In this section, we present an extensive evaluation of Craft, the implementation of our abstraction framework and the CH-Zonotope domain, on monDEQs using multiple architectures and datasets including CIFAR10 [Krizhevsky et al. 2009], MNIST [LeCun et al. 1998], and HCAS [Julian and Kochenderfer 2019]. First, we evaluate Craft in the setting of local robustness certification against the challenging ℓ_∞-perturbations (MNIST and CIFAR10). There, we demonstrate that Craft outperforms the current state-of-the-art in scalability, speed, and precision. Second, we show in the HCAS setting that Craft is also suitable for deriving global guarantees. Third, we investigate the impact of different algorithmic components in an ablation study. Finally, we demonstrate Craft's broader applicability on a numerical program.

Experimental Setup. We implement Craft in PyTorch [Paszke et al. 2019] and evaluate it on single Nvidia TITAN RTX using a 16-core Intel Xeon Gold 6242 CPU at 2.80GHz. For implementation and experimental details as well as (hyper)parameter choices, please see the detailed description and full code in our artifact.

Implementation Details. The version of Algorithm 1 presented here is slightly simplified for the sake of clarity. These implementation details, however, impact neither the soundness of the algorithm nor the intuitions outlined here.

6.1 Local Robustness Certification with Craft

Similar to prior work [Chen et al. 2021], we evaluate the first 100 test set samples and report the mean runtime for correctly classified samples (Time), the certified accuracy (Cert.), and the number of samples for which we found an abstract post-fixpoint (Cont.).

In Table 2, we show results for a range of fully connected and convolutional monDEQs. There, *#Bound* denotes the number of samples empirically robust to PGD attacks [Madry et al. 2018] (20 restarts using 5 output diversification [Tashiro et al. 2020] and 50 margin loss steps [Gowal et al. 2019]) and constitutes an upper bound to the certified accuracy. We generally observe that smaller fully-connected networks have lower empirical robustness but are easier to certify, with the smallest network yielding the highest certified accuracy. Surprisingly, we find that on both MNIST and CIFAR10, convolutional networks are comparatively easy to verify, yielding the highest certified accuracies.

Table 3. Comparison of Craft to the 'Robustness Model' (SemiSDP) of Chen et al. [2021].

Model	*Latent Size*	*# Acc.*	*ϵ*	*# Bound*	SemiSDP		Craft (ours)	
					# Cert.	Time [s]	# Cert.	Time [s]
`FCx40`	40	99	0.01	98	**98**	401.5	**98**	**0.97**
			0.02	95	88	357.7	**94**	**8.82**
			0.05	70	18	196.4	**36**	**17.19**
			0.07	29	5	121.0	**8**	**21.25**
			0.10	10	0	63.0	0	12.88
`FCx87`	87	99	0.01	99	98	1388.1	**99**	**1.40**
			0.02	98	92	1186.8	**98**	**2.66**
			0.05	75	24	599.9	**30**	**15.75**
			0.07	42	**5**	387.6	**5**	**14.53**
			0.10	8	0	214.46	0	9.75

Comparison with SemiSDP. Chen et al. [2021] introduce three models suitable for robustness certification. In Table 3, we compare against the (by far) most precise of these approaches, the 'Robustness Model' (SemiSDP), which is the current state-of-the-art for verifying ℓ_∞ robustness properties for monDEQs. As the underlying SDP solver limits SemiSDP to MNIST networks with a latent space size of at most 87 neurons [Chen et al. 2021], we compare to them only on our two smallest networks: their `FCx87` and our `FCx40`. For the smallest perturbations of $\epsilon = 0.01$, both tools are able to certify (almost) all empirically robust samples, with SemiSDP failing to certify one sample on `FCx87`. However, while Craft requires only around 1s per sample, SemiSDP takes three to four orders of magnitude longer (401.5s and 1388.1s). For larger perturbation magnitudes $\epsilon \in \{0.02, 0.05, 0.07\}$, Craft is consistently more precise and much faster, certifying up to 100% more samples (36 vs 18 for `FCx40` at $\epsilon = 0.05$) with around two orders of magnitude shorter average runtime. For $\epsilon = 0.1$, few samples are empirically robust and neither tool can verify robustness for any on either network. Finally, as shown in Table 2, Craft scales to much larger networks (10x) and more challenging datasets (CIFAR10) than SemiSDP. The two alternative certification models proposed by Chen et al. [2021], the 'Lipschitz Model' and the 'Ellipsoid Model', are significantly less precise, verifying no property at all for $\epsilon = 0.05$ and `FCx87`. Thus we omit a detailed comparison.

Comparison with Lipschitz-Bound-Based Methods. Three existing works derive Lipschitz-Bounds for monDEQs, either via a posteriori analysis [Chen et al. 2021; Pabbaraju et al. 2021] or construction [Revay et al. 2020]. However, they all obtain significantly lower certified accuracies.

6.2 Global Robustness Certification with Craft

To demonstrate that Craft is also suitable for computing global robustness certificates, we analyze the HCAS (Horizontal Collision Avoidance System) setting which is illustrated in Fig. 10 and has been used as a benchmark for feed-forward networks in the past [Fu and Li 2021; Julian and Kochenderfer 2019]. Given the relative position (x- and y-coordinates) and heading (ϑ) of an intruder aircraft (shown in red) with respect to one's own position and heading (shown in black), one of five action recommendations (COC - Clear of Conflict, WL/WR - Weak Left/Right, SL/SR - Strong Left/Right) is given. The training data is generated by framing this as a Markov Decision Process (MDP) and solving it for many parameters, yielding a large look-up table (see Julian and Kochenderfer [2019] for more details). We train a monDEQ (`FCx100`) on this large and discrete dataset to obtain a continuous and compressed mapping.

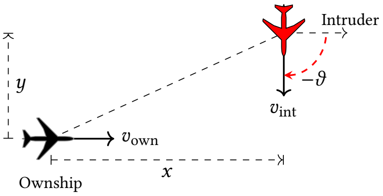
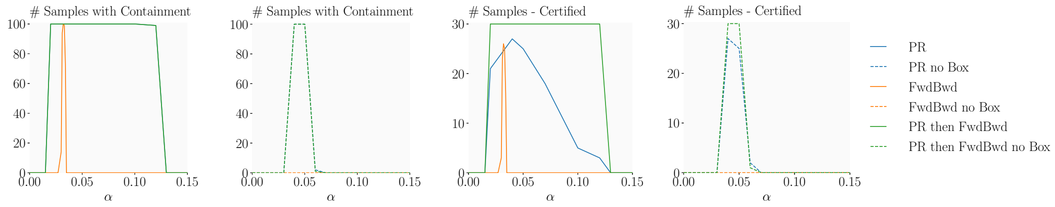


Fig. 10. Visualization of the HCAS Geometry. Adapted from Julian and Kochenderfer [2019].



(a) Containment (b) Containment w/o box (c) Certification (d) Certification w/o box

Fig. 12. Illustration of the stability ranges for α, depending on the use of the box component, and fixpoint solver. Note that the blue (PR) and green (PR then FB) lines are identical in the two containment plots (left).

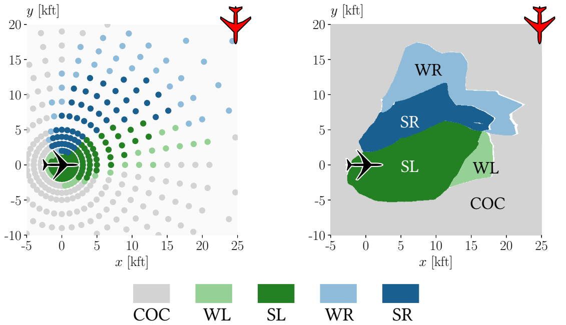


Fig. 11. HCAS policy training data (left) for ($\vartheta = -90°$) and verified monDEQ prediction (right) ($\vartheta \in [-90.5°, -89.5°]$). The colored regions are certified to yield the indicated recommendation. No certificate for the white regions.

To confidently use this monDEQ representation, we aim to certify that it yields consistent predictions across large regions of the input space. Using CRAFT, we apply a domain splitting approach [Wang et al. 2018] in order to exhaustively certify decisions for the whole input space.This way, we can certify the prediction on 82.8% of the relevant input region. For visualization, we pick a thin slice of this space and visualize the resulting certified decision regions (right) and the corresponding tabular data (left) in Fig. 11. There, regions for which we obtain a certificate are colored depending on the action recommended, and regions for which no certificate is obtained are shown in white. We observe that, as expected, the regions directly at the decision boundary can not be certified. However, we also observe a small unexpected pocket of non-certifiable decisions where a strong right is certifiably recommended all around.

6.3 Ablation Study on CRAFT

We conduct an extensive ablation study on the key features of CRAFT and report results for FCx87 in Table 4 and Fig. 12.

CH-Zonotope. We analyze the effectiveness of our domain by setting either $\boldsymbol{b} = 0$ (no Box) or $\boldsymbol{A} = \boldsymbol{0}$ (no Zono). Disallowing the Zonotope component leaves a standard Box, which converges quickly, but fails to prove any property (see Table 4). Disallowing the Box component, leaving a CH-Zonotope which still utilizes error consolidation rather than a standard Zonotope, can yield the same precision but significantly reduces the range of dampening parameters α leading to convergence (compare Fig. 12a and Fig. 12b), to the point where for some solvers and networks (e.g. FB and FCx87) we were unable to find such an α.

Table 4. Overview of the natural accuracy (*Acc.*), the number of samples for which the fixpoint set iteration converged (*Cont.*) the certified accuracy (*Cert.*), and the average time per sample on FCx87.

Ablation	*# Cont.*	*# Cert.*	*Time [s]*
Reference	100	**30**	17.48
No Zono component	100	0	0.38
No Box component	100	**30**	23.18
Only PR	100	27	**4.10**
Only FB	100	26[†]	7.99
No λ optimization	100	24	7.81
Reduced λ optimization	100	27	13.85
Same iter. containment	100	0	7.14
No Expansion	50	9	18.89

[†] No formal guarantee as conditions for Theorem 3.1 are not satisfied.

Iteration Method. As discussed in Section 5.2, we can choose different operator splitting methods for the containment-finding and the tightening phase of Craft. When using only FB, the α range for which we can detect containment is extremely narrow (see Fig. 12a) and does not overlap the region $0 < \alpha < 2m/\|\boldsymbol{I} - \boldsymbol{W}\|_2^2 = 0.0125$ for which we have convergence guarantees in the concrete. This is problematic, as these guarantees are a condition for Theorem 3.1 and thus our formal soundness guarantee. Using PR until we find containment and then FB avoids this issue, is significantly more robust to the choice of α, and yields the tightest abstractions of all three approaches, leading to the most certified properties (see Fig. 12c). Only using PR leads to slightly less precise final abstractions and thus worse certification performance. First using FB and then PR is not supported by Theorem 3.3, as we would not have computed $\hat{\mathcal{U}}^*$. We thus use first PR and then FB, for all other experiments. While we fix α_1 for PR, we choose α_2 for FB adaptively.

Transformer Optimization. Recall that the abstract ReLU transformer has a parametrizable slope λ, which can be optimized to tighten our final abstractions [Weng et al. 2018; Wong and Kolter 2018; Zhang et al. 2018] by unrolling several iterations of the solver and using (projected) gradient descent to optimize λ individually for each of these iterations. We distinguish three settings, 'No λ optimization', 'Reduced λ optimization', and 'Reference', where we unroll no, 20, and 40 iterations and optimize lambda over no, 60, and 200 gradient steps, respectively. We only perform this optimization for samples that are already close to being certified, allowing us to certify six additional samples while only increasing the mean certification time by 10s (see Table 4).

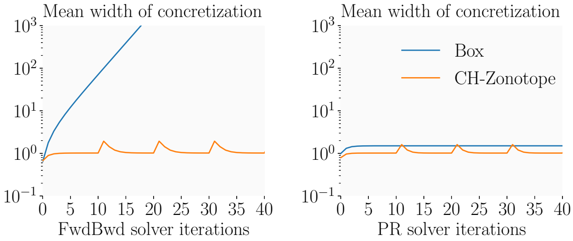
Same Iteration Containment. To demonstrate the value of fixpoint set preservation (Definition 3.2 and Theorems 3.3 and 5.1), we consider the setting 'Same iter. containment', where we always require the abstraction $\hat{\mathcal{S}}_{n+1}$ that we use to certify the postcondition to be contained in its predecessor $\hat{\mathcal{S}}_n$. In this setting, we are not able to certify a single property (see Table 4), as we can only detect strictly smaller abstractions if the current abstraction is still relatively loose.

Expansion. To illustrate the effect of artificially expanding our abstractions as part of error consolidation (see Section 5.2), we consider 'No Expansion' in Table 4, where we turn expansion off by setting w_{mul} and w_{add} to 0. Most notably, for 50% of samples, we do not detect abstraction containment and thus do not obtain sound fixpoint abstractions at all. Further, even if we obtain fixpoint abstractions, we often do not certify the corresponding sample.

6.4 Effectiveness of CH-Zonotope

Here, we evaluate the effectiveness of our novel CH-Zonotope.

Precision. We compare CH-Zonotope to Box, the only other domain commonly used in neural network verification that enables a tractable containment check (see Table 1). In Fig. 13, we show the mean width of the concretized abstractions as a proxy for the domain's precision over the number of abstract solver iterations for a representative sample. Empirically, we find that the Box domain is significantly less precise, diverging quickly when using FB splitting and being too imprecise to prove any property when using PR splitting. For CH-Zonotope, we observe how error consolidation periodically simplifies the abstraction, increasing its size, before additional solver applications tighten it again.

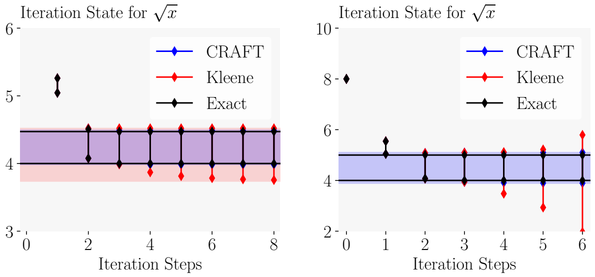


(a) $g_\alpha^{FB\#}$ (b) $g_\alpha^{PR\#}$

Fig. 13. Mean width of concretizations over the solver iteration for a representative sample on FCx40.

```
def root(x):
    s = s₀
    while s ≤ 0 or |s ∗ s − 1/x| ≥ ϵ:
        h = (1 − x ∗ s ∗ s)
        s = s + s ∗ (0.5 ∗ h + 0.375 ∗ h ∗ h)
    return s
```

Fig. 14. Program root, which calculates the square root of input x through iterative Householder approximation.



(a) $\mathcal{X} = [16, 20]$

(b) $\mathcal{X} = [16, 25]$

Fig. 15. Comparison of root ($1/s_i$) over-approximations for different intervals $\mathcal{X}$. We show the final interval as shaded region.

6.5 Case Study: Analysis of Square Root Approximation

In this section, we provide a simple example of the wider applicability of our abstract interpretation approach of fixpoint iterations and its advantages compared to Kleene iteration.

We consider the Householder method to compute (the reciprocal of) square roots, illustrated in Fig. 14 and commonly used as a test case [Ghorbal et al. 2009; Goubault et al. 2007]. We consider the input set $\mathcal{X} = [16, 20] \subseteq \mathbb{R}$ (with exact fixpoint set $[4, \sqrt{20} \approx 4.472]$), the initialization $s_0 = 2^{-3}$, and the termination threshold $\epsilon = 10^{-8}$. We use the Zonotope domain [Ghorbal et al. 2009] and compare to Kleene iteration with semantic unrolling [Blanchet et al. 2002], i.e., we iterate $\hat{\mathcal{S}}_i = f^{\#}(\hat{\mathcal{S}}_{i-1})$ if we can show the termination condition to not be satisfied and else $\hat{\mathcal{S}}_i = \hat{\mathcal{S}}_{i-1} \sqcup f^{\#}(\hat{\mathcal{S}}_{i-1})$. Using Kleene iteration, we thus obtain the fixpoint set abstraction $\hat{\mathcal{S}}^*_{\text{kl}}$, shown in red in Fig. 14a, which contains all intermediate iteration states for which the termination condition might trigger. Our abstract interpreter Craft allows us to, instead, compute iterations as $\hat{\mathcal{S}}_i = f^{\#}(\hat{\mathcal{S}}_{i-1})$ until our contraction-based termination condition triggers ($\hat{\mathcal{S}}_i \sqsubseteq \hat{\mathcal{S}}_{i-1}$). This yields a more precise fixpoint set over-approximation $\hat{\mathcal{S}}^*_{\text{cr}}$, shown in blue in Fig. 14a. Further, if we consider a more challenging precondition of $\mathcal{X} = [16, 25]$, Kleene iteration quickly diverges (see Fig. 14b), while Craft computes the precise abstraction $\hat{\mathcal{S}}^*_{\text{cr}}$ (see Table 5). Note that Craft requires 10 and 18 iterations for $\mathcal{X} = [16, 20]$ and $\mathcal{X} = [16, 25]$, respectively, while Kleene iteration requires 30 for $\mathcal{X} = [16, 20]$. We show truncated versions of the iteration in Fig. 15 for readability.

Table 5. Comparison of the fixpoint over-approximations obtained with different methods. Exact mathematical fixpoints (Exact), their over-approximation obtained via Craft and Kleene iteration.

Method		Root Interval $1/\gamma(\mathcal{S}^*)$	
		$\mathcal{X}$ = [16,20]	$\mathcal{X}$ = [16,25]
Exact	$\mathcal{S}^*$	[4.000, 4.472]	[4.000, 5.000]
Craft	$\hat{\mathcal{S}}^*_{\text{cr}}$	[3.983, 4.493]	[3.887, 5.104]
Kleene iteration	$\hat{\mathcal{S}}^*_{\text{kl}}$	[3.738, 4.520]	[0.000, ∞)

6.6 Limitations

From the existence of (global) Lipschitz bounds on monDEQs [Pabbaraju et al. 2021] and the convergence guarantees in the concrete, it follows that an exact abstract iteration converges to bounded fixpoint sets for every bounded input region. However, Craft has some limitations that could prevent us from computing them: (i) Our termination criterion (Theorem 4.2) requires a contraction of the abstract iteration state. However, this contraction is not guaranteed to occur, even when using exact abstractions, and might not be detected by our incomplete containment check, even if it does occur. (ii) While an exact abstract iteration is guaranteed to converge, there is no such guarantee for its over-approximation, which could diverge due to imprecisions accumulated by the use of incomplete abstract transformers. Despite these limitations, which we share with conventional Kleene iteration, we observe empirically that we can find fixpoint set approximations in all evaluated cases when using PR splitting (see Table 2).

7 RELATED WORK

We now briefly review related work on Zonotope order reduction, neural network verification, and abstract interpretation.

Zonotope Order Reduction. While the flexibility and expressiveness of Zonotope have made it a popular abstract domain for safety and reachability analysis [Althoff et al. 2008; Kühn 1998; Yang and Scott 2018], its representation size can grow quickly. To alleviate this limitation, Kühn [1998] suggested soundly over-approximating Zonotope using smaller, less precise representations, i.e., fewer error terms and thus smaller error matrices. This is called order-reduction via outer-approximation. While a range of such methods was introduced in the following years [Combastel 2003; Girard 2005; Yang and Scott 2018; Yazarel and Pappas 2004], they were designed for Zonotope in very ($p \leq 10$) or relatively ($p \leq 100$) low dimensional spaces and of low order ($k \leq 50$). Thus, they generally scale poorly to the larger dimensions ($p \geq 500$) and high orders ($k \geq 1000$) we consider [Kopetzki et al. 2017]. In this setting, the PCA-based method proposed by Kopetzki et al. [2017], which we build on, was found to produce the tightest approximations while still being computationally tractable.

Incomplete Neural Network Verification. Incomplete verification approaches (such as ours) are generally fast and efficient but sacrifice precision, i.e., they may fail to certify properties that do hold. They can be divided into bound propagation [Gehr et al. 2018; Singh et al. 2018, 2019b; Xu et al. 2020; Zhang et al. 2018] and optimization problem based approaches, using linear programming (LP) [Ferrari et al. 2022; Müller et al. 2022; Singh et al. 2019a] or semidefinite programming (SDP) formulations [Raghunathan et al. 2018]. However, existing approaches are unable to handle (unbounded) fixpoint iterations and thereby (mon)DEQ verification without non-trivial adaptations.

In contrast to the above deterministic approaches, which analyze models as they are, stochastic defenses such as randomized smoothing [Cohen et al. 2019; Lecuyer et al. 2018] construct new robust models by introducing noise into the inference process. They establish robustness guarantees for these new models with high probability but incur significant runtime costs at both certification- and inference-time. This drawback is further exacerbated by the relatively expensive fixpoint iterations needed in (mon)DEQ inference.

Certification of monDEQs. Two main approaches have been proposed to certify the robustness of monDEQs: (i) Pabbaraju et al. [2021] use the special structure of monDEQs to bound the global Lipschitz constant of the network, and (ii) Chen et al. [2021] adapt an SDP-based approach by introducing a semi-algebraic representation of the ReLU-operator used in monDEQs.

While the global Lipschitz bounds from Pabbaraju et al. [2021] do not require a per-sample analysis, they are inherently loose, especially in the challenging setting of ℓ_∞ perturbations, where our approach achieves much higher precision, as demonstrated in Section 6.

Depending on the encoding, the SDP-encoding by Chen et al. [2021] allows to bound the score difference between classes, the global Lipschitz constant, or yields an ellipsoidal relationship between inputs and outputs. All three approaches only scale to an implicit layer size of 87 neurons due to the limitations of the underlying SDP solver [Chen et al. 2021]. Additionally, the most effective approach suffers from long runtimes (up to 1400s per sample) even for these small networks, making the certification of many inputs or larger networks infeasible. We compare favorably to this approach in terms of precision, runtime, and scalability in Section 6.

Orthogonally, Revay et al. [2020] show a way of bounding the Lipschitz constant of a monDEQ by construction but do not report any robustness certificates. Further, enforcing small Lipschitz constants this way reduces the resulting accuracy significantly, thus limiting the utility of the obtained networks.

Recently, Wei and Kolter [2022] built on the work from Revay et al. [2020] by further restricting the parametrization of monDEQs to make them more amenable to verification with the Box domain and thus permitting larger models with higher accuracy to be analyzed. However, for the general monDEQs we consider, their method reduces to analysis with the Box domain, which we found to be too imprecise to prove any property.

Abstract Interpretation of Unbounded Loops. Abstract interpreters employ many techniques to either speed up the analysis of unbounded loops or make it more precise [Goubault et al. 2007]. These approaches, include semantic unrolling [Blanchet et al. 2002], widening, and narrowing [Amato et al. 2016; Bourdoncle 1993; Cousot and Cousot 1977a,b, 1992b]. Gange et al. [2013] discuss considerations for Kleene iteration on non-lattice abstract domains such as CH-Zonotope.

8 CONCLUSION

We presented a novel abstract interpretation approach for fixpoint iterators with convergence guarantees based on two key contributions: (i) theoretical insights which allow us to compute sound and precise fixpoint abstractions without using joins, and (ii) a new abstract domain, CH-Zonotope, which allows for precise propagation of abstract elements and enables efficient inclusion checks ($O(p^3)$ in dimension p). To demonstrate the effectiveness of this approach, we implemented our framework in a tool called Craft and evaluated it on the robustness verification of monDEQs, a novel neural architecture constituting a particularly challenging instance of a high-dimensional fixpoint iterators.

In an extensive evaluation, we demonstrated that Craft exceeds state-of-the-art performance in monDEQ verification by two orders-of-magnitude in terms of speed, one order of magnitude in terms of scalability, and about 25% in terms of certification rate, demonstrating the merit of our framework.

9 ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their constructive comments, our colleague Florian Dorner for his insightful feedback, and our Shepherd Swarat Chaudhuri.

This work has been done as part of the EU grant ELSA (European Lighthouse on Secure and Safe AI, grant agreement No. 101070617) and the SERI grant SAFEAI (Certified Safe, Fair and Robust Artificial Intelligence, contract no. MB22.00088). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the European Commission can be held responsible for them. The work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI).

10 FURTHER RESOURCES

We have published all code, models, and instructions required to reproduce our results on Zenodo [Müller et al. 2023] with an updated version being available at github.com/eth-sri/craft.

For a more in-depth empirical analysis, especially of the CH-Zonotope, we refer the interested reader to the extended version of this work at https://arxiv.org/abs/2110.08260.

REFERENCES

Matthias Althoff, Olaf Stursberg, and Martin Buss. 2008. Verification of uncertain embedded systems by computing reachable sets based on zonotopes. *IFAC Proceedings Volumes* 41, 2 (2008).

Gianluca Amato and Francesca Scozzari. 2012. The Abstract Domain of Parallelotopes. *Electron. Notes Theor. Comput. Sci.* 287 (2012). https://doi.org/10.1016/j.entcs.2012.09.003

Gianluca Amato, Francesca Scozzari, Helmut Seidl, Kalmer Apinis, and Vesal Vojdani. 2016. Efficiently intertwining widening and narrowing. *Sci. Comput. Program.* 120 (2016). https://doi.org/10.1016/j.scico.2015.12.005

Brandon Amos and J. Zico Kolter. 2017. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *Proc. of ICML*, Vol. 70.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2019. Deep Equilibrium Models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada.*

Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. 2002. Design and Implementation of a Special-Purpose Static Program Analyzer for Safety-Critical Real-Time Embedded Software. In *The Essence of Computation, Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones [on occasion of his 60th birthday]*, Vol. 2566. https://doi.org/10.1007/3-540-36377-7_5

François Bourdoncle. 1993. Efficient chaotic iteration strategies with widenings. In *Formal Methods in Programming and Their Applications, International Conference, Akademgorodok, Novosibirsk, Russia, June 28 - July 2, 1993, Proceedings*, Vol. 735. https://doi.org/10.1007/BFb0039704

Tong Chen, Jean-Bernard Lasserre, Victor Magron, and Edouard Pauwels. 2021. Semialgebraic Representation of Monotone Deep Equilibrium Models and Applications to Certification. *ArXiv preprint* abs/2106.01453 (2021).

Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. 2019. Certified Adversarial Robustness via Randomized Smoothing. In *Proc. of ICML*, Vol. 97.

Christophe Combastel. 2003. A state bounding observer based on zonotopes. In *2003 European Control Conference (ECC)*. IEEE.

Patrick Cousot and Radhia Cousot. 1977a. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977.* https://doi.org/10.1145/512950.512973

Patrick Cousot and Radhia Cousot. 1977b. Static Determination of Dynamic Properties of Recursive Procedures. In *Formal Description of Programming Concepts: Proceedings of the IFIP Working Conference on Formal Description of Programming Concepts, St. Andrews, NB, Canada, August 1-5, 1977.*

Patrick Cousot and Radhia Cousot. 1979. Constructive versions of Tarski's fixed point theorems. *Pacific journal of Mathematics* 82, 1 (1979).

Patrick Cousot and Radhia Cousot. 1992a. Abstract Interpretation Frameworks. *J. Log. Comput.* 2, 4 (1992). https://doi.org/10.1093/logcom/2.4.511

Patrick Cousot and Radhia Cousot. 1992b. Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation. In *Programming Language Implementation and Logic Programming, 4th International Symposium, PLILP'92, Leuven, Belgium, August 26-28, 1992, Proceedings*, Vol. 631. https://doi.org/10.1007/3-540-55844-6_142

Claudio Ferrari, Mark Niklas Müller, Nikola Jovanovic, and Martin T. Vechev. 2022. Complete Verification via Multi-Neuron Relaxation Guided Branch-and-Bound. In *Proc. of ICLR.*

Feisi Fu and Wenchao Li. 2021. Sound and Complete Neural Network Repair with Minimality and Locality Guarantees. *ArXiv preprint* abs/2110.07682 (2021).

Graeme Gange, Jorge A. Navas, Peter Schachte, Harald Søndergaard, and Peter J. Stuckey. 2013. Abstract Interpretation over Non-lattice Abstract Domains. In *Static Analysis - 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013. Proceedings*, Vol. 7935. https://doi.org/10.1007/978-3-642-38856-9_3

Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA.* https://doi.org/10.1109/SP.2018.00058

Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Y. Tsai. 2021. Implicit Deep Learning. *SIAM J. Math. Data Sci.* 3, 3 (2021). https://doi.org/10.1137/20M1358517

Khalil Ghorbal, Eric Goubault, and Sylvie Putot. 2009. The Zonotope Abstract Domain Taylor1+. In *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, Vol. 5643. https://doi.org/10.1007/978-3-642-02658-4_47

Antoine Girard. 2005. Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control.* Springer.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *Proc. of ICLR.*

Eric Goubault and Sylvie Putot. 2008. Perturbed affine arithmetic for invariant computation in numerical program analysis. *CoRR* abs/0807.2961 (2008). arXiv:0807.2961

Eric Goubault, Sylvie Putot, Philippe Baufreton, and Jean Gassino. 2007. Static Analysis of the Accuracy in Control Systems: Principles and Experiments. In *Formal Methods for Industrial Critical Systems, 12th International Workshop, FMICS 2007, Berlin, Germany, July 1-2, 2007, Revised Selected Papers*, Vol. 4916. https://doi.org/10.1007/978-3-540-79707-4_3

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. 2018. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. *ArXiv preprint* abs/1810.12715 (2018).

Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy A. Mann, and Pushmeet Kohli. 2019. An Alternative Surrogate Loss for PGD-based Adversarial Testing. *ArXiv preprint* abs/1910.09338 (2019).

Dwight Guth. 2013. A formal semantics of Python 3.3. (2013).

Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. 2020. Faster Dynamic Matrix Inverse for Faster LPs. *ArXiv preprint* abs/2004.07470 (2020).

Kyle D Julian and Mykel J Kochenderfer. 2019. Guaranteeing safety for neural network-based aircraft collision avoidance systems. In *Digital Avionics Systems Conference (DASC)*.

Kai Kellner. 2015. Containment problems for projections of polyhedra and spectrahedra. *ArXiv preprint* abs/1509.02735 (2015).

Suyong Kim, Weiqi Ji, Sili Deng, Yingbo Ma, and Christopher Rackauckas. 2021. Stiff neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31, 9 (2021).

Anna-Kathrin Kopetzki, Bastian Schürmann, and Matthias Althoff. 2017. Methods for order reduction of zonotopes. In *56th IEEE Annual Conference on Decision and Control, CDC 2017, Melbourne, Australia, December 12-15, 2017*. https://doi.org/10.1109/CDC.2017.8264508

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

Wolfgang Kühn. 1998. Rigorously computed orbits of dynamical systems without the wrapping effect. *Computing* 61, 1 (1998).

Adrian Kulmburg and Matthias Althoff. 2021. On the co-NP-completeness of the zonotope containment problem. *European Journal of Control* (2021).

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998).

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2018. Certified Robustness to Adversarial Examples with Differential Privacy. *2019 IEEE Symposium on Security and Privacy (S&P)* (2018).

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proc. of ICLR*.

Matthew Mirman, Timon Gehr, and Martin T. Vechev. 2018. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In *Proc. of ICML*, Vol. 80.

Mark Niklas Müller, Marc Fischer, Robin Staab, and Martin Vechev. 2023. *Abstract Interpretation of Fixpoint Iterators with Applications to Neural Networks - Artifact*. https://doi.org/10.5281/zenodo.7794269

Mark Niklas Müller, Gleb Makarchuk, Gagandeep Singh, Markus Püschel, and Martin Vechev. 2022. PRIMA: General and Precise Neural Network Certification via Scalable Convex Hull Approximations. *Proc. ACM Program. Lang.* 6, POPL, Article 43 (2022), 33 pages. https://doi.org/10.1145/3498704

Chirag Pabbaraju, Ezra Winston, and J. Zico Kolter. 2021. Estimating Lipschitz constants of monotone deep equilibrium models. In *Proc. of ICLR*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*.

Sylvie Putot. 2012. Static analysis of numerical programs and systems. *Habilitation à diriger des recherches, Université de Paris-Sud* (2012).

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*.

Max Revay, Ruigang Wang, and Ian R. Manchester. 2020. Lipschitz Bounded Equilibrium Networks. *ArXiv preprint* abs/2010.01732 (2020).

Ernest K Ryu and Stephen Boyd. 2016. Primer on monotone operator methods. *Appl. Comput. Math* 15, 1 (2016).

Sadra Sadraddini and Russ Tedrake. 2019. Linear Encodings for Polytope Containment Problems. In *58th IEEE Conference on Decision and Control, CDC 2019, Nice, France, December 11-13, 2019*. https://doi.org/10.1109/CDC40024.2019.9029363

François Serre, Christoph Müller, Gagandeep Singh, Markus Püschel, and Martin Vechev. 2021. Scaling Polyhedral Neural Network Verification on GPUs. In *Proc. Machine Learning and Systems (MLSys)*.

Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin T. Vechev. 2019a. Beyond the Single Neuron Convex Barrier for Neural Network Certification. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. 2018. Fast and Effective Robustness Certification. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. 2019b. An abstract domain for certifying neural networks. *PACMPL* 3, POPL (2019). https://doi.org/10.1145/3290354

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proc. of ICLR*.

Yusuke Tashiro, Yang Song, and Stefano Ermon. 2020. Output Diversified Initialization for Adversarial Attacks. *ArXiv preprint* abs/2003.06878 (2020).

Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. 2019. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proc. of ICML*, Vol. 97.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Formal Security Analysis of Neural Networks using Symbolic Intervals. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*.

Colin Wei and J Zico Kolter. 2022. Certified Robustness for Deep Equilibrium Models via Interval Bound Propagation. In *International Conference on Learning Representations*.

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. In *Proc. of ICML*, Vol. 80.

Ezra Winston and J. Zico Kolter. 2020. Monotone operator equilibrium networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Eric Wong and J. Zico Kolter. 2018. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *Proc. of ICML*, Vol. 80.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. 2020. Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xuejiao Yang and Joseph K Scott. 2018. A comparison of zonotope order reduction techniques. *Automatica* 95 (2018).

Hakan Yazarel and George J Pappas. 2004. Geometric programming relaxations for linear system reachability. In *Proceedings of the 2004 American Control Conference*, Vol. 1. IEEE.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. 2018. Efficient Neural Network Robustness Certification with General Activation Functions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*.

Received 2022-11-10; accepted 2023-03-31